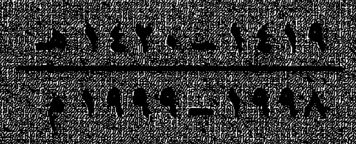
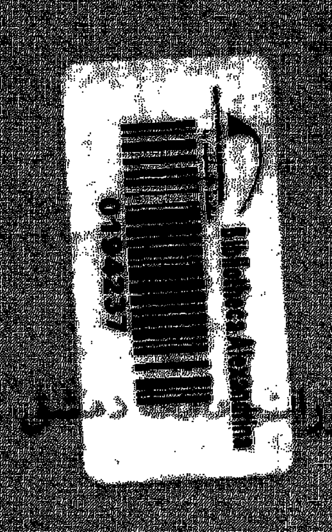
الدكتور نور الدين شيخ عبيد | الدكتور محمد نوار العوا | الدكتور محمد علوان | الدكتور عماد القابوني

بنيان الحواسيب والمعالجات الصغرية

الطبعة الأولى





الدكتور
نور الدين شيخ عبيد

الدكتور
محمد نوار العوا

الدكتور
مجد علوان

الدكتور
عماد الصابوني

بنيان الحواسيب والمعالجات الصغرية

الطبعة الأولى

١٤١٩ - ١٤٢٠ هـ
١٩٩٨ - ١٩٩٩ م

منشورات جامعة دمشق

كلمة المؤلفين

يهدف هذا الكتاب إلى تقديم المفاهيم الأساسية في بنيان الحواسيب والمعالجات الصغرية، أي إعطاء الطالب صورة واضحة عن البنية «المادية» للحاسوب، التي تكوّن، مع البنية البرمجية، إحدى الدعامتين اللتين تقوم عليهما علوم المعلوماتية.

بُني الحاسوب منذ أجياله الأولى على فكرة أساسها ذاكرة يُحفظ فيها البرنامج ومعالج يُنفّذ هذا البرنامج سطراً بعد سطر. وماتزال هذه الفكرة مسيطرة على بنى الحواسيب التقليدية حتى اليوم، وإن ظهرت بُنىً، نسميها بنىً تفرعية، تقوم بتنفيذ عدة مهمات برمجية في الوقت نفسه، مؤدية إلى إنقاص مدة التنفيذ، وهو كسب هام في بعض التطبيقات، مثل فك الرسائل المشفرة أو مراقبة حركة إعصار...

بيد أن التطور الذي حصل، منذ الأجيال الأولى للحواسيب، كان في الواقع تطوراً في التقانة، وتمثّل أولاً في اختراع الترانزستور في الخمسينيات، الذي حل محل المصابيح المتعددة المساري؛ ثم في تصنيع الدارات المتكاملة في الستينيات، التي ضمت في بداياتها عشرات الترانزستورات، وتضم اليوم الملايين منها في رقاقات لاتزيد مساحتها عن بضعة ملّيمترات مربعة. وجاء أخيراً تصميم المعالج الصغري في بداية السبعينيات الذي قاد إلى ثورة في عالم المعلوماتية تمثلت في ظهور الحاسوب الشخصي الذي تترافق أجياله وأجيال المعالج الصغري منذ ذاك الوقت.

زادت سرعة التنفيذ في أجيال المعالجات وتطورت التطبيقات المنفَّذة بواسطتها، فأصبح بالإمكان مثلاً تنفيذ عمليات حسابية كانت تحتاج إلى برنامج كامل في أجيال المعالجات الأولى، مثل التقسيم الحسابي والقيمة المطلقة... كذلك زاد عدد الخانات الاثنانية التي تتعامل معها المعالجات في لحظة ما، من أربع خانات في البدايات الأولى إلى أربع وستين خانة (أو أكثر) حالياً. وهناك من يتحدث اليوم عن معالجات فوتونية ومعالجات كمومية نأمل منها أن تزيد أكثر من سرعة المعالج.

يتمحور كتابنا هذا، وهو مرجع الطالب في مادة «فيزياء الحواسيب»، حول المعالج الصغري، نواة النظم الحاسوبية الشائعة. والكتاب هو حصيلة خبرتنا التدريسية في المعهد العالي للعلوم التطبيقية والتكنولوجيا، وينقسم إلى جزأين وملاحق يُرجع إليها عند الحاجة. يعالج الجزء الأول موضوع الدارات المنطقية التي تكوّن اللبنات الأساسية المستخدمة في بناء المعالج الصغري، في حين يُخصص الجزء الثاني للمعالجات الصغرية وبنيان الحواسيب.

يتناول الجزء الأول الجبر الاثناني والدوال المنطقية وأشكالها المبسطة، وكذلك الدارات المنطقية الأساسية والدارات المنطقية التركيبية الشهيرة التي يعتمد خرجها في لحظة ما على دخلها في تلك اللحظة. ننتقل بعد ذلك إلى معالجة الدارات التتابعية، التي يعتمد خرجها في لحظة ما على دخلها في تلك اللحظة وعلى ما تعرضت له مداخل هذه الدارات في الماضي. ونعالج في نهاية هذا الجزء الدارات التتابعية المتزامنة، أي التي تعمل على إيقاع «ساعة»، تقوم مقام ضابط الإيقاع في الفرقة الموسيقية.

أما في الجزء الثاني، فندرس المعالجات الصغرية، ونأخذ مثالاً عليها المعالج الشهير INTEL 8086، الذي قامت على أجياله المختلفة الحواسيب الشخصية؛ ونتطرق إلى الوحدات المحيطية المرتبطة بالمعالج، التي تختص بنقل المعلومات بين العالم الخارجي والمعالج.

يلي ذلك دراسة برمجة المعالج 8086 بلغة المجمّع، وهي اللغة التي يفهمها المبرمج والمعالج في الوقت نفسه. ونعرض أخيراً لمحة إلى المعالجات المتطورة في مجالي الحواسيب التفرعية والحواسيب ذات مجموعة التعليمات الموجزة.

أما الملاحق، فيُخصص أولها للتذكرة بأنظمة العد والترميز؛ أما ثانيها فيهدف إلى تقديم فكرة موجزة عن تقانات تنفيذ وتحقيق المؤثرات المنطقية أو البوابات المنطقية؛ على حين يصف الملحق الثالث الشكل الخارجي للمعالج 8086 ويشرح وظائف مرابطه؛ ونقدم في الملحق الرابع مجموعة تعليمات المعالج 8086 التي يمكن الرجوع إليها كدليل للاستخدام. ونختتم كتابنا بمقترح لمواضيع الجلسات العملية التي ترافق الدروس النظرية.

نود هنا أن نتقدم بخالص الشكر لزملائنا في المعهد العالي لما قدموه من مساعدة وملاحظات، ونخص بالذكر الدكتور مروان زبيبي لمتابعته الحثيثة لتقدم عملنا، والدكتور حسام الدين الفوال لما بذله من جهد في مسح الأشكال. وكذلك نشكر الآنسة إيمان الصواف التي بذلت مجهوداً كبيراً، تميز بالدقة والكفاءة، في إدخال المخطوط اليدوي إلى الحاسوب؛ ونشكر كذلك السيد رفيع السوادي لجهده في التدقيق الأولي لعملية الإدخال إلى الحاسوب.

نرجو أن نكون قد وفقنا في عملنا؛ ونأمل بمساعدة القراء، من أساتذة وطلاب، الوصول بهذا الكتاب في الطبعات القادمة إلى وضع أفضل، وتجاوز ما قد يكون فيه من هنات وثغرات.

دمشق في ٢ شباط ١٩٩٩

المؤلفون

المحتويات

الجزء الأول: الدارات المنطقية

الجزء الثاني: المعالجات الصغرية وبنيان الحواسيب

الملاحق

الملحق الثاني: التنفيذ التقاني للمؤثرات المنطقية 307



الجزء الأول

الدارات المنطقية



الفصل الأول

تذكرة بالجبر الاثناني

1 مقدمة

يكون الحكم في قضايا عديدة بأحد أمرين: إما «الإيجاب» أو «النفي»، ويأخذ هذا الحكم أشكالاً مختلفة. فناتج عملية حسابية هو «صواب» أو «خطأ»؛ وشرط النجاح هو «محقق» أو «غير محقق»؛ وباب السيارة هو «مغلق» أو «مفتوح»؛ والاتصال بين حاسوب وآخر هو «قائم» أو «غير قائم»... ويعبر عادة عن ذلك رياضياً باعطاء حكم الإيجاب القيمة '1' وحكم النفي القيمة '0'.

2 جبر بول الاثناني

لتكن المجموعة $A = \{0,1\}$ مجموعة قيم المتغيرات المنطقية [1]، ولنعرف عليها العمليات التالية:

- الجمع المنطقي:

جمع متغيرين a,b (ويشار إليه بـ a+b) عملية نتيجتها 1 إذا كان a أو b (أو كلاهما) مساوياً للواحد؛ ونتيجتها 0 إذا كان كلا المتغيرين مساوياً للصفر. نلاحظ أن الصفر لا يؤثر في ناتج العملية، وهو من ثم

1 انظر الملحق الأول للتذكرة بأنظمة العد والترميز.

عنصر حيادي بالنسبة إلى الجمع.

- الضرب المنطقي:

ضرب متغيرين a,b (ويشار إليه بـ a.b أو ab) عملية نتيجتها 1 إذا كان المتغيران معاً مساويين للواحد. للضرب إذن عنصر حيادي هو الواحد.

للعملتين السابقتين الخواص التالية:

- التجميعية:

$a+(b+c) = (a+b) +c$
$a.(b.c) = (a.b).c$

- التبديلية:

$a+b =b+a$
$a.b = b.a$

- التوزيعية:

الضرب بالنسبة إلى الجمع:

$a.(b+c) = a.b + a.c$

الجمع بالنسبة إلى الضرب:

$a + c.b = (a+c).(a+b)$

يمكن البرهنة على العلاقات السابقة مباشرة بمناقشة كافة الحالات الممكنة. (لنلاحظ أن الخاصة الأخيرة لا نجدها إلا في الجبر الاثناني.)

- التتميم:

لكل عنصر a متمم وحيد (نرمز له بـ $\overline{a}$ أو ¬a) يحقق ما يلي:

$\overline{a} + a = 1$
$a . \overline{a} = 0$

تكوّن المجموعة A والعمليات + و . والتتميم جبراً نسميه بجبر

بول Boole الاثناني، ونرمز إليه بـ $B = <\{0,1\}, +, ., ->$.

3 قانونا دو مورغان De Morgan

نصادف في كثير من الأحيان صيغاً تضم عملية نفي الضرب أو عملية نفي الجمع. العملية الأولى تساوي جمع النفي أما العملية الثانية فتساوي جداء النفي. يعبر عن ذلك قانونا دومورغان:

الأول : $\overline{a.b} = \overline{a} + \overline{b}$

الثاني : $\overline{a+b} = \overline{a}.\overline{b}$

يمكن البرهنة على صحة القانونين السابقين إما جبرياً أو باستخدام جدول الحقيقة لكل طرف من طرفي العلاقة والتوثق من تطابقهما (انظر 5 أدناه).

4 الدوال والصيغ المنطقية

الدوال والصيغ المنطقية هي دوال مجموعة المنطلق فيها $\{0,1\}^p$ ومجموعة المستقر $\{0,1\}$، كالدالة التالية:

$m(a,b,c) = a.b.\overline{c} + a.\overline{b}.c + \overline{a}.b.c + a.b.c$

حيث a,b,c متغيرات منطقية.

تسمى هذه الدالة بدالة الأكثرية، ذلك لأنها تكون محققة إذا كان متغيران على الأقل من المتغيرات الثلاثة محققين.

مثال آخر على الدوال المنطقية دالة لاغرانج ذات المتحولات الثلاثة a,b,c التي تأخذ قيمة b إذا كان a محققاً وتأخذ قيمة c إذا كان a غير محقق. يعبّر عن ذلك بالشكل التالي:

$U(a,b,c) = a.b + \overline{a}.c$

5 جدول الحقيقة لدالة منطقية

يمكن أن نعدّ، لدالة بـ n متغيراً منطقياً، جدولاً بمدخلين مؤلفاً من

2^n خانة تقابل الأعمدة فيه مختلف تراكيب بعض المتغيرات في حين تقابل الأسطر مختلف تراكيب المتغيرات الباقية. يوضع في كل خانة من خانات الجدول القيمة 1 إذا كانت الدالة محققة والقيمة 0 في الحالة المعاكسة.

لنضرب مثالاً على ذلك دالة الأكثرية:

$$m(a,b,c) = a.b.\overline{c} + a.\overline{b}.c + \overline{a}.b.c + a.b.c$$

التي لها جدول الحقيقة التالي:

c \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

حيث تأخذ الدالة القيمة 1 إذا كان اثنان من المتغيرات مساويين للواحد.

6 المطابقات الشهيرة

نصادف في كثير من الدوال البولانية تعابيرَ أو حدوداً متكافئة بالرغم من الاختلاف الظاهر في شكل صيغها. وفيما يلي أكثر تلك الصيغ شيوعاً، والتي تسمى بالمطابقات الشهيرة:

$a + 1 = 1,\ a + 0 = a,\ a.1 = a,\ a.0 = 0$

$a + a = a,\ a.a = a$

$a + a.b = a,\ a + \overline{a}.b = a + b,\ (a + \overline{b}).b = a.b$

$\overline{a}.b + a.\overline{b} = (a + b).(\overline{a} + \overline{b}),\ \overline{\overline{a}.b + a.\overline{b}} = (a+\overline{b}).(\overline{a}+b)$

يمكن البرهنة على التطابق مباشرة، أو بالاستعانة بجدول الحقيقة لكل طرف والتوثق من التطابق.

7 الدوال المنطقية

1-7 الدوال المنطقية الشهيرة بمتغيرين

يمكن إيجاد ست عشرة دالة بمتغيرين بولانيين a,b، ذلك أن لجدول الحقيقة في هذه الحالة أربع خانات، وكل خانة يمكن أن تأخذ القيمة 0 أو القيمة 1. سنذكر فيما يلي أهم تلك الدوال:

- النفي $f1 = \overline{a}$، ويشار إليها بـ NOT.
- الجمع $f2 = a + b$، ويشار إليها بـ OR.
- نفي الجمع $f3 = \overline{a + b}$، لذا يشار إليها بـ NOR.
- الضرب $f4 = a.b$، ويشار إليها بـ AND.
- نفي الضرب $f5 = \overline{a.b}$، لذا يشار إليها بـ NAND.
- الجمع الاثناني التام $f6 = \overline{a}.b + a.\overline{b}$، ويشار إليها بـ XOR.

تسمي الدوال الشهيرة NOT, OR, AND, NOR, NAND, XOR بالمؤثرات المنطقية. وتُستخدم في التأثير في متغيرين أو أكثر.

2-7 الدوال المنطقية بأكثر من متغيرين

لتكن المتغيرات المنطقية x1, x2, x3, ... xn. يمكن تعريف عمليات الضرب والجمع المنطقية كما في حالة متغيريين، مع الاستفادة من كون كلتا العمليتين تجميعيتين:

$$\sum_{i=1}^{n} xi = x1 + x2 + ... + xn$$

$$\prod_{i=1}^{n} xi = x1.x2.x3. ... .xn$$

يُعمم قانونا دومورغان كما يلي:

$$\overline{\sum_{i=1}^{n} xi} = \overline{x1}\,.\,\overline{x2}\,.\,\overline{x3}\,.\; \ldots .\overline{xn}$$

$$\overline{\prod_{i=1}^{n} xi} = \overline{x1} + \overline{x2} + \overline{x3} + \ldots + \overline{xn}$$

يمكن كتابة أي دالة منطقي بأحد شكلين: مجموع جداءات (الشكل المنفصل)، أو جداء مجاميع (الشكل المتصل).

- الشكل المنفصل:

في هذا الشكل، تُكتب الدالة على شكل مجموع جداءات (Sum Of Product) بين المتغيرات المنطقية بشكلها الصحيح أو المتمم، كما في المثال التالي:

$f(x1,x2,x3) = x1.\overline{x2} + x1.x2.\overline{x3}$

- الشكل المتصل:

في هذا الشكل، تُكتب الدالة على شكل جداء مجاميع (Product Of Sum) بين المتغيرات المنطقية بشكلها الصحيح أو المتمم، كما في المثال التالي:

$g(x1,x2,x3) = (\overline{x1}+x2).(\overline{x1}+\overline{x2}+x3)$

نسمي حداً في الدالة المنطقية أيّ جداء للمتغيرات المنطقية بشكليها الصحيح أوالمتمم في حالة الشكل المنفصل، كالحد $x1.x2.\overline{x3}$ ؛ أو أيّ مجموع للمتغيرات المنطقية بشكليها الصحيح أوالمتمم في حالة الشكل المتصل، كالحد $\overline{x1}+\overline{x2}+x3$.

8 الشكل القانوني لدالة منطقية

وهو الشكل الذي تظهر في كل حد من حدوده كافة المتغيرات المنطقية بأحد الشكلين الصحيح أو المتمم.

<u>مثال</u>: للدالتين المذكورين آنفاً الشكل القانوني التالي:

$$f(x1,x2,x3) = x1.\overline{x2}.x3 + x1.\overline{x2}.\overline{x3} + x1.x2.\overline{x3}$$

$$g(x1,x2,x3) = (\overline{x1}+x2+x3).(\overline{x1}+x2+\overline{x3}).(\overline{x1}+\overline{x2}+x3)$$

وتتساوى دالتان منطقيتان إذا كان لهما الشكل القانوني نفسه.

9 مجموعات المؤثرات

1-9 مجموعات المؤثرات التامة

نقول عن مجموعة من المؤثرات المنطقية Σ إنها تامة، إذا سمحت بالوصول إلى مجمل العمليات المنطقية الأساسية من جمع وضرب وتتميم $\Omega = \{+, ., \neg\}$.

فالمجموعة $\Sigma1= \{+, \neg\}$ هي مجموعة تامة، ذلك أنه يمكن استنتاج الضرب من الجمع والتتميم على النحو التالي:

$$\overline{\overline{a}+\overline{b}} = a.b$$

نلاحظ أن المجموعة السابقة هي مجموعة NOR، ومن ثم يمكن تمثيل أية دالة منطقية باستخدام نوع واحد فقط من المؤثرات هو NOR.

والمجموعة $\Sigma2 = \{., \neg\}$ هي كذلك مجموعة تامة، ذلك أنه يمكن استنتاج الجمع من الضرب والتتميم على النحو التالي:

$$\overline{\overline{a}.\overline{b}} = a+b$$

نلاحظ أن المجموعة السابقة هي مجموعة NAND، ومن ثم يمكن تمثيل أية دالة منطقية باستخدام المؤثر NAND فقط.

أما المجموعة $\Sigma3 = \{+, .\}$ فهي ليست تامة، إذ لا يمكن استنتاج التتميم من الضرب والجمع.

<u>تمرين</u>: *هل المجموعة* $\{\oplus, .\}$ *تامة؟* (العملية $\oplus$ ترمز لـ XOR.)

2-9 مجموعات المؤثرات التامة والصغرى

نقول عن مجموعة مؤثرات تامة Σ إنها صغرى إذا لم يكن ممكناً إيجاد مجموعة جزئية منها Σ1 تامة وعدد عناصرها أقل من عدد عناصر المجموعة Σ. فالمجموعة {¬ ,. ,+} تامة ولكنها ليست صغرى، لأن كلتا المجموعتين الجزئيتين {¬ ,+} و {¬ ,.} تامتان. ولكن هاتين المجموعتين تامتان وكل منهما مجموعة صغرى، إذ لا يمكن اشتقاق مجموعة جزئية تامة من أيٍ منهما.

تمرين: *هل المجموعة* {. ,⊕} صغرى؟

10 التمثيل الصندوقي للمؤثرات المنطقية

نستخدم عند رسم المخططات المنطقية تمثيلاً متعارفاً عليه على شكل «رموز صندوقية» يمكّن بالنظر إليها من معرفة كل مؤثر، ومعرفة الدالة المنطقية التي تحققها جملة من المؤثرات المترابطة. يطلق على الرمز الصندوقي (وعلى المكوّن المادي الذي يحقق المؤثر [2]) تسمية البوابة المنطقية Logical Gate.

يمكن لهذه البوابات (ما عدا مؤثر التتميم) أن تكون ذات مدخلين أو أكثر، ولكن عدد المداخل يكون محدوداً عادة لأسباب تقانية.

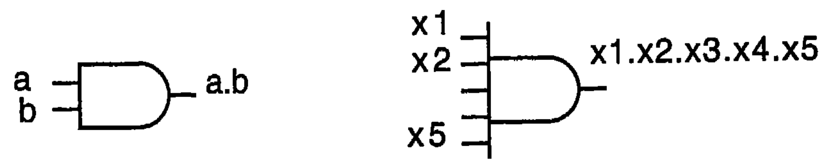
- مؤثر التتميم أو النفي أو بوابة NOT:

a —▷o— $\bar{a}$

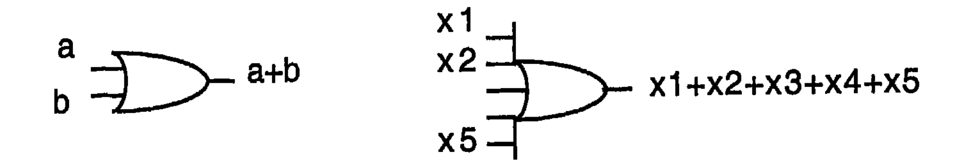
2 انظر الملحق الثاني من أجل عرض سريع للتنفيذ التقاني للمؤثرات المنطقية.



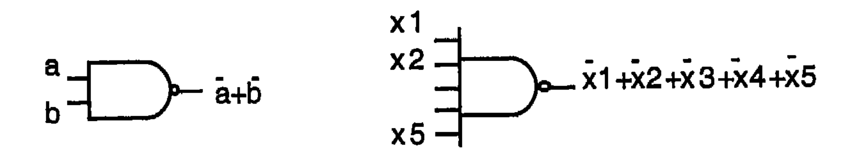
- بوابة الضرب أو بوابة AND:



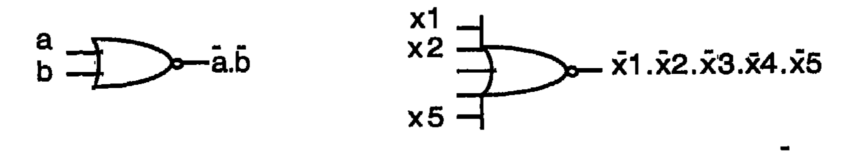
- بوابة الجمع أو بوابة OR:



- بوابة الضرب المتمم أو بوابة NAND:



- بوابة الجمع المتتم أو بوابة NOR:



• بوابة الجمع التام أو بوابة XOR:

a, b → $\bar{a}.b + a.\bar{b}$

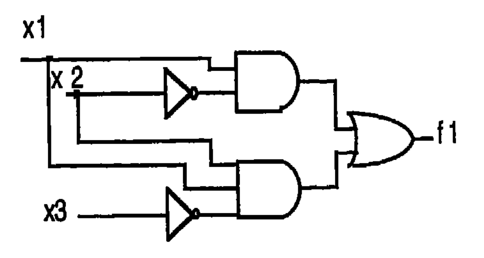
والتي يمكن أن توجد أيضاً بعدد من المداخل أكبر من اثنين.

يمكننا باستخدام هذه الرموز تمثيل أي دالة منطقية.

مثال: لتكن الدالة المنطقية:

$$f1(x1,x2,x3) = x1.\overline{x2} + x1.x2.\overline{x3}$$

التي تمثل بالمخطط المنطقي التالي:



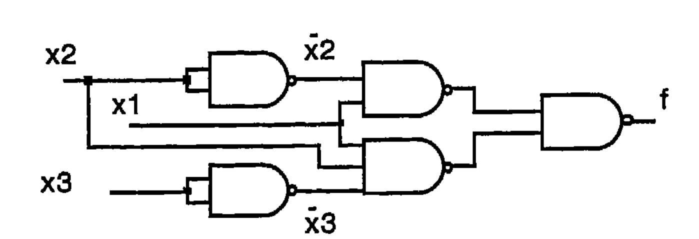
نلاحظ إمكان تنفيذ هذه الدالة باستخدام بوابات منطقية من نوع NAND فقط، وذلك بعد ملاحظة أن:

$$f1(x1,x2,x3) = \overline{\overline{x1.\overline{x2} + x1.x2.\overline{x3}}}$$

وبالتالي:

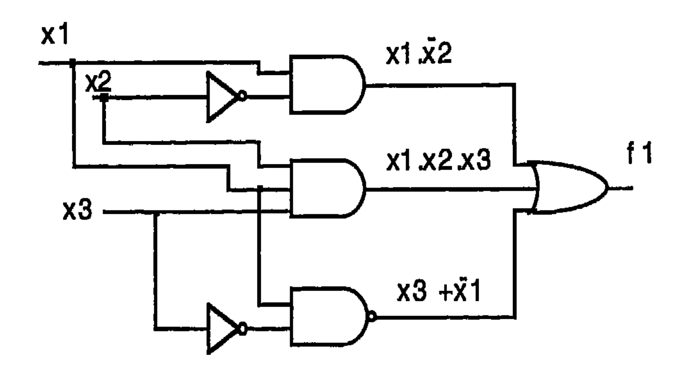
$$f1(x1,x2,x3) = \overline{\overline{x1.\overline{x2}}\ .\ \overline{x1.x2.\overline{x3}}}$$

وبملاحظة أن $\overline{x} = \overline{x.x}$، تُمثّل الدالة السابقة كما يلي:



وبالعكس، يمكن إيجاد علاقة الدالة المنطقية إذا أعطي المخطط الصندوقي. من أجل ذلك، نقوم بتحديد المداخل وإعطاء كل منها اسماً، ثم بكتابة الحدود الناتجة على المخارج، وأخيراً بتحديد الصيغة النهائية للدالة.

مثال:



يمكن من المخطط السابق إيجاد علاقة الدالة المنطقية التي تساوي:

$$f(x1,x2,x3) = x1.\overline{x2} + x1.x2.x3 + \overline{x1} + x3$$



الفصل الثاني

تبسيط الدوال المنطقية

1 مقدمة

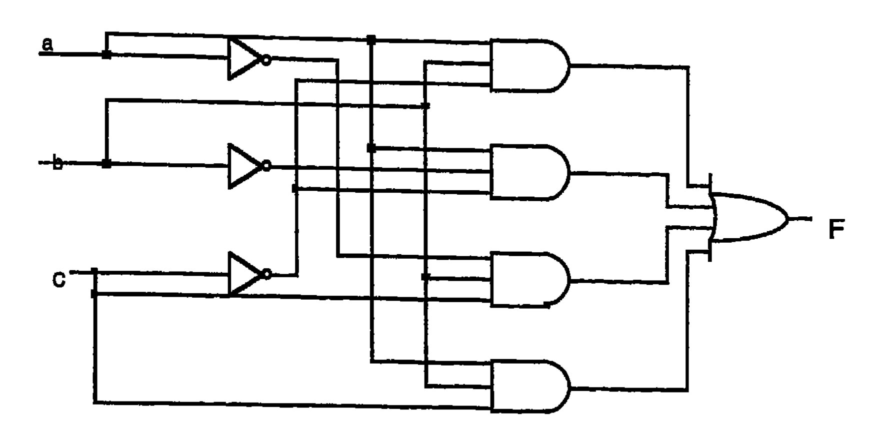
المقصود بتبسيط الدوال المنطقية إيجاد صيغة مكافئة للدالة باستخدام عدد أقل من المتغيرات والحدود، وذلك بهدف تخفيض كلفة التنفيذ التي تتناسب طرداً مع عدد البوابات وعدد التوصيلات. تكون الصيغة المبسطة للدالة أحياناً سهلة المنال، ويمكن إيجادها بمجرد الاعتناء بصياغة المسألة. لنأخذ مثالاً على ذلك دالة الأكثرية (التي نذكر أنها تكون محققة إذا كان اثنان من متغيراتها الثلاثة على الأقل محققين والثالث غير محقق بالضرورة):

$F(a,b,c) = a.b.\overline{c} + a.\overline{b}.c + \overline{a}.b.c + a.b.c$

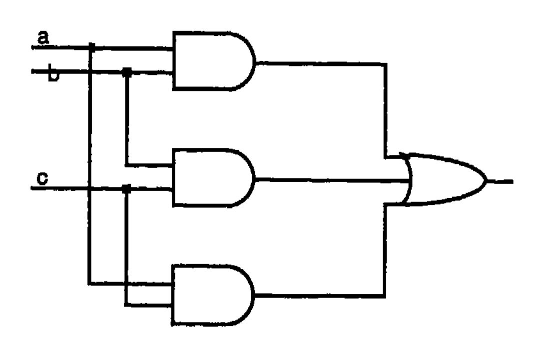
حيث أُخذت في الصيغة السابقة المتغيرات الثلاثة بأوضاعها المختلفة. ولكن الوصول إلى النتيجة المطلوبة ممكن أيضاً بأخذ متغيرين محققين معاً، مما يعطي الشكل المكافئ التالي:

$F(a,b,c) = a.b + b.c + a.c$

والصيغة الأخيرة هي وضوحاً أبسط صيغة؛ لأن تحقيق الدالة بشكلها الأول يقود إلى دارة منطقية لها المخطط التالي:



في حين أن الشكل الثاني يمكن تحقيقه بدارة لها المخطط التالي:



نلاحظ أن المخطط الثاني يحتاج إلى عدد أقل من البوابات وخطوط التوصيل، بالمقارنة بالمخطط الأول؛ فقد احتجنا في الحالة الأولى إلى 3 عواكس و 4 بوابات AND وبوابة OR و 19 خط توصيل، أما في الحالة الثانية فلا نحتاج إلا إلى 3 بوابات AND وبوابة OR و 9 خطوط توصيل، وفي هذا اقتصاد واضح في الكلفة.

ولكن كيف يمكن الحصول على أبسط صيغة لدالة ما؟ الأمر ليس دائما بسهولة المثال السابق.

2 الصيغة الصغرى لدالة منطقية

يستند حل مسألة إيجاد الصيغة الصغرى لدالة منطقية إلى فكرة

حذف المتغيرات المنطقية غير الأساسية، والتي نبينها فيما يلي.

ليكن التركيب المنطقي m(x1,x2,...,xn) المكوّن من مجموع حدين مكتوبين بالشكل القانوني التالي:

$$m(x1, x2, ..., xn) = g(x1, x2, ... xi\text{-}1, xi\text{+}1, ... xn).xi + g(x1, x2, ... xi\text{-}1, xi\text{+}1, ... xn).\overline{x}i$$

حيث x1, x2, ... xn متغيرات منطقية تنتمي قيمها إلى المجموعة {0,1}. عندئذ يكون التركيب m مكافئاً للتركيب g ويكون المتغير xi غير أساسي في التركيب m.

وفي حال إمكان كتابة التركيب m كجداء حدين على الشكل التالي:

$$m(x1, x2, ... xn) = [g(x1, x2, ... xi\text{-}1, xi\text{+}1, xn) + xi].[g(x1, x2, ... xi\text{-}1, xi\text{+}1, xn) + \overline{x}i]$$

يكون التركيب m مكافئاً للتركيب g ويكون المتغير xi غير أساسي.

1-2 إيجاد الصيغة الصغرى باستخدام المطابقات الشهيرة

للحصول على الصيغة الصغرى لدالة منطقية ما، نبحث عن المتغيرات غير الأساسية بين التراكيب المكوّنة لها واختزالها بالشكل المبين آنفاً؛ وتكون الدالة بصيغتها الصغرى إذا كانت كافة المتغيرات الداخلة في صيغتها أساسية بالنسبة إلى مختلف تراكيب حدودها.

عند البحث عن الصيغة الصغرى لدالة منطقية يمكن الاستفادة من العلاقات الشهيرة المعروفة في المنطق الاثناني، وأهمها:

$a + \overline{a} = 1,\ a.\overline{a} = 0$

$a + 1 = 1,\ a + 0 = a,\ a.1 = a,\ a.0 = 0$

$a + a = a,\ a.a = a$

$a+b.c = (a+b)(a+c),\ a(b+c) = a.b+a.c$

$a + a.b = a,\ a + \overline{a}.b = a + b,\ (a + \overline{b}).b = a.b$

$\overline{a}.b + a.\overline{b} = (a + b).(\overline{a} + \overline{b}),\ \overline{\overline{a}.b + a.\overline{b}} = (a+\overline{b}).(\overline{a}+b)$

<u>مثال</u>: لتكن دالة الأكثرية:

$$F(a,b,c) = a.b.\overline{c} + a.\overline{b}.c + \overline{a}.b.c + a.b.c$$

نلاحظ أن المتغير c غير أساسي بالنسبة إلى الحدين الأول والرابع، من ثم يمكن أن نستبدل بهما a.b؛ وكذلك الأمر فيما يتعلق بالمتغير b بين الحد الثاني والحد الرابع، من ثم يمكن أن نستبدل بهما a.c؛ وأخيراً المتغير a ليس أساسياً بالنسبة للحدين الأخيرين اللذين يستعاض عنهما بـ b.c. وبهذا نحصل على الصيغة المبسطة التالية:

$$F(a,b,c) = a.b + a.c + b.c$$

نلاحظ أنه لا توجد متغيرات أخرى غير أساسية بين أزواج حدود الدالة بشكلها الأخير، من ثم فهذه الصيغة هي الصيغة الصغرى.

<u>مثال</u>: لتكن الدالة المنطقية:

$$F(a,b,c,d) = \overline{a}.\overline{b}.\overline{c}.\overline{d} + \overline{a}.\overline{b}.c.\overline{d} + a.\overline{b}.\overline{c}.\overline{d} + a.\overline{b}.c.\overline{d}$$

بأخذ الحدين الأول والثاني معاً، والحدين الثالث والرابع معاً نجد:

$$F(a,b,c,d) = \overline{a}.\overline{b}.\overline{d} + a.\overline{b}.\overline{d}$$

وبملاحظة أن a غير أساسي، نجد أخيراً:

$$F(a,b,c,d) = \overline{b}.\overline{d}$$

<u>تمرين</u>: برهن أن الصيغة الصغرى للدالة:

$$F(a,b,c,d) = \overline{a}\,\overline{b}\,\overline{c}\,\overline{d} + \overline{a}\,\overline{b}\,\overline{c}d + \overline{a}b\overline{c}\,\overline{d} + \overline{a}b\overline{c}d + \overline{a}bcd + a\overline{b}c\overline{d}$$

هي:

$$F(a,b,c,d) = \overline{a}.\overline{b} + \overline{a}.b.d + a.\overline{b}.c.\overline{d}$$

لنلاحظ صعوبة استخدام هذه الطريقة في حالة الدوال ذات المتغيرات العديدة والحدود الكثيرة، إضافة إلى كونها تحتاج إلى مران دائم. ولكن الأهم هو عدم وجود أداة تسمح لنا بالتوثق من صحة النتيجة. لذا جرى البحث عن طريق أخرى أسهل استخداماً. وهذا هو موضوع الفقرات اللاحقة.

2-2 إيجاد الصيغة الصغرى باستخدام مخططات كارنو

تعتمد طريقة كارنو Karnaugh على مبدأ التجاور، حيث تكون الكلمة m والكلمةg، المؤلفة كل منهما من n خانة ثنائية، متجاورتين إذا أمكن الانتقال من m إلى g بتغيُّر خانة واحدة فقط من 0 إلى 1 أو من 1 إلى 0.

<u>مثال:</u>

الكلمة m = 000110 مجاورة للكلمة g = 000111 وهي لا تجاور الكلمة f = 000110.

تتلخص طريقة كارنو في إنشاء جدول حقيقة للدالة المنطقية المكتوبة بالشكل القانوني، توزع على طرفيه اليساري والعلوي المتغيرات الخاصة بالدالة المنطقية المدروسة بحيث تأخذ فيه المتغيرات قيماً متجاورة.

لنأخذ كمثال على ذلك دالة الأكثرية في غير صيغتها الصغرى:

$F(a,b,c) = a.b.\overline{c} + a.\overline{b}.c + \overline{a}.b.c + a.b.c$

ولننشئ جدولاً أعمدته تقابل قيم التراكيب المتجاورة للمتغيرين a,b، وأسطره تقابل قيمتي المتغير c؛ ولنضع في خانات الجدول القيم التي تأخذها الدالة من أجل المتغيرات المقابلة للخانة موضع الاهتمام:

c \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

بالطريقة نفسها يمكن تمثيل الدالة:

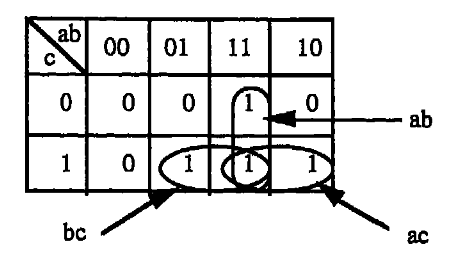
$F(a,b,c,d) = \overline{a}.\overline{b}.\overline{c}.\overline{d} + \overline{a}.\overline{b}.c.\overline{d} + a.\overline{b}.\overline{c}.\overline{d} + a.\overline{b}.c.\overline{d}$

بالجدول التالي:

cd \ ab	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

لنلاحظ أن التجاور في قيم المتغيرات ab أو cd يبقى مستمراً بين الخانة الأولى والثانية، ثم الثانية والثالثة، ثم الثالثة والرابعة، ثم الرابعة والأولى. ويعني وجود القيمة 1 في خانتين متجاورتين شاقولياً أو أفقياً أن المتغيرالمنطقي الموافق، الذي يأخذ القيمتين الصفر والواحد، هو غير أساسي ويمكن اختصاره من صيغة الدالة. على هذا نبحث، لإيجاد الصيغة الصغرى لدالة منطقية بإستخدام مخططات كارنو، عن التجاورات الممكنة بشكلها الأوسع.

لنأخذ مثالاً على ذلك دالة الأكثرية كما أعطيت في مخطط كارنو سابقاً، فالتجاورات الممكنة هي المبينة في الشكل التالي:



c \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

لذا فالحدود الأساسية هي: a.b, a.c, b.c.

والصيغة الصغرى هي:

$$F(a,b,c) = a.b + a.c + b.c$$

<u>مثال</u>: لتكن الدالة المنطقية:

$$F(a,b,c,d) = \overline{a}.\overline{b}.\overline{c}.\overline{d} + \overline{a}.\overline{b}.c.\overline{d} + a.\overline{b}.\overline{c}.\overline{d} + a.\overline{b}.c.\overline{d}$$

التي يظهر الجدول التالي التجاورات الممكنة فيها:

ab / cd	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

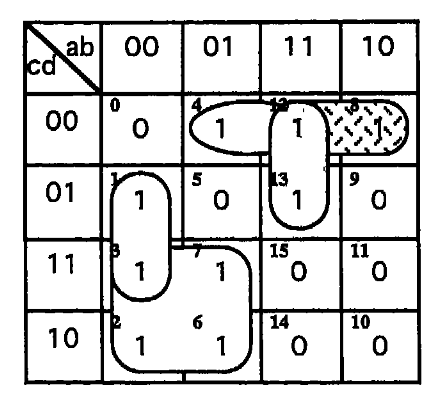
نلاحظ هنا أن الزوايا الأربع متجاورة مثنى مثنى، ومن السطرين الأول والأخير نجد أن المتغير a ليس أساسياً، وكذلك من العمود ين الأول والأخير نجد أن المتغير c ليس أساسياً، ومن ثم فالصيغة الصغرى للدالة هي:

$F(a,b,c,d) = bd$

<u>مثال</u>: لتكن الدالة المنطقية:

$F(a,b,c,d) = \bar{a}b\bar{c}\bar{d} + ab\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}cd + \bar{a}\bar{b}c\bar{d} + \bar{a}bcd + \bar{a}bc\bar{d}$

يبيّن الجدول التالي جدول كارنو لهذه الدالة وكذلك كيفية تشكيل التجاورات اللازمة للحصول على الصيغة الصغرى.



ab / cd	00	01	11	10
00	(0) 0	(4) 1	(12) 1	(8) 1
01	(1) 1	(5) 0	(13) 1	(9) 0
11	(3) 1	(7) 1	(15) 0	(11) 0
10	(2) 1	(6) 1	(14) 0	(10) 0

لنعط كل حد في الجدول رقماً يقابل القيمة العددية لمتغيراته بالترتيب abcd.

الحدود المتجاورة هي:

- 4 تجاور 12 حيث a هو متغير غير أساسي؛
- 12 تجاور 8 حيث b غير أساسي؛
- 1 تجاور 3 حيث c غير أساسي؛
- 3 و 7 و 6 و 2 متجاورة حيث b و d غير أساسيين؛
- 12 تجاور 13 حيث d غير أساسي.

نلاحظ أن الحد 12 اشترك في عدة تجاورات. كما نلاحظ أن الحدود 4 و 12 و 8 لا تكوّن تجاوراً واحداً، ذلك أن 4 تجاور12 و 12 تجاور 8 ولكن 4 لا تجاور 8.

على هذا فالصيغة الصغرى للدالة هي:

$$F(a,b,c,d) = \bar{a}.c + \bar{a}.\bar{b}.d + a.b.\bar{c} + a.\bar{c}.\bar{d} + b.\bar{c}.\bar{d}$$

ويجب الانتباه إلى عدم جدوى أخذ الحدين 6 و 7 في تجاور إضافي، إذ إن هذين الحدين قد شاركا في تجاور أوسع.

<u>مثال</u>: لتكن الدالة المنطقية ذات المتغيرات الخمسة a,b,c,d,e:

$$F(a,b,c,d,e) = \bar{a}\bar{b}\bar{c}\bar{d}\bar{e} + \bar{a}\bar{b}\bar{c}\bar{d}e + \bar{a}b\bar{c}\bar{d}\bar{e} + \bar{a}b\bar{c}\bar{d}e + a\bar{b}\bar{c}d\bar{e} + a\bar{b}\bar{c}de + a\bar{b}cd\bar{e} + a\bar{b}cde$$
$$+ \bar{a}\bar{b}\bar{c}de + \bar{a}\bar{b}cde + \bar{a}\bar{b}c\bar{d}e + \bar{a}b\bar{c}d\bar{e} + \bar{a}bcd\bar{e} + \bar{a}bc\bar{d}\bar{e} + ab\bar{c}d\bar{e} + abcd\bar{e}$$

نحتاج هنا إلى جدول ذي ثمانية أعمدة وأربعة أسطر بغية البحث عن الصيغة الصغرى. وتجنباً لصعوبة التعامل مع جدول بهذا الحجم، نلجأ إلى استخدام جدولين بأربعة أعمدة يتعامل كل منهما مع أربعة من المتغيرات، ولتكن a,b,c,d، ونخصص أحد الجدولين لـ e = 1 والآخر لـ e = 0، فالخانات المتطابقة في كلا الجدولين تكوّن تجاوراً يكون فيه المتغير e غير أساسي. هذان الجدولان في مثالنا هذا هما:

e=0

cd \ ab	00	01	11	10
00	1	1		
01		1	1	1
11		1	1	1
10		1		

e=1

cd \ ab	00	01	11	10
00	1	1		
01	1			1
11	1			1
10	1			

يوضح الشكل السابق التجاورات الممكنة في كل من الجدولين وفي كليهما معاً، ومنه نجد أن الصيغة الصغرى لهذه الدالة هي:

$$F(a,b,c,d,e) = \overline{a}.\overline{c}.\overline{d} + \overline{a}.b.\overline{e} + a.d.\overline{e} + a.\overline{b}.d + \overline{a}.\overline{b}.e$$

وجهنا اهتمامنا حتى الآن إلى إيجاد الصيغة الصغرى باستخدام جداول كارنو، وذلك بمعالجة الخانات التي تأخذ فيها الدالة القيمة 1، للحصول على الصيغة على شكل مجموع حدود. ولكن ذلك ليس بالحل الأفضل دائماً، على الأقل من حيث عدد البوابات المنطقية أو عدد خطوط التوصيل اللازمة لتنفيذ الصيغة الصغرى. لنأخذ على سبيل المثال الدالة المنطقية ذات جدول الحقيقة التالي:

cd \ ab	00	01	11	10
00	1	0	1	1
01	0	0	1	0
11	0	0	0	0
10	1	0	0	0

والتي يمكن، بمعاينة التجاورات المبينة على الجدول، الحصول على صيغتها الصغرى التالية:

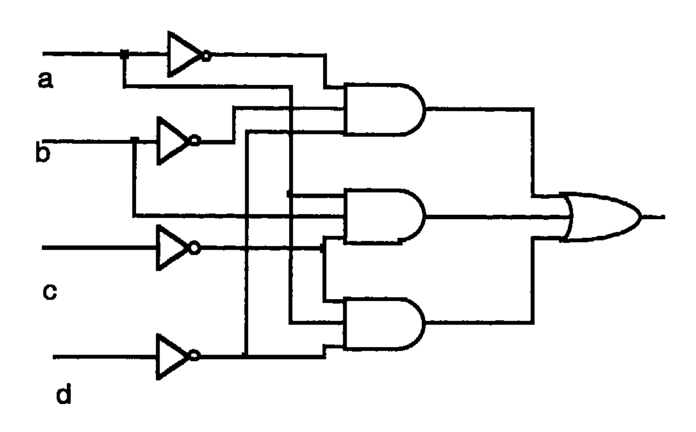
$F(a,b,c,d) = \overline{a}.\overline{b}.\overline{d} + a.b.\overline{c} + a.\overline{b}.\overline{d}$

لو أخذنا الآن الخانات التي تحوي أصفاراً فقط، ثم أخذنا متمم الناتج لكان لدينا:

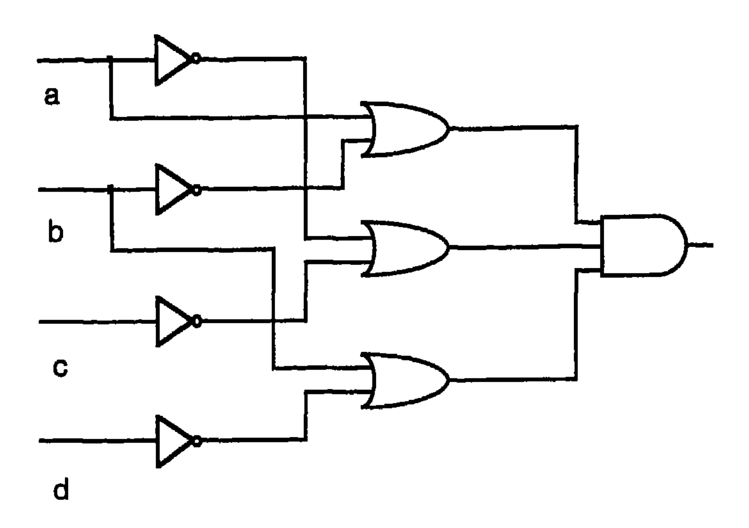
cd \ ab	00	01	11	10
00	1	0	1	1
01	0	0	1	0
11	0	0	0	0
10	1	0	0	0

$F(a,b,c,d) = (a + \overline{b}).(\overline{a} + \overline{c}).(b + \overline{d})$

لا تختلف هذه الصيغة عن الصيغة السابقة إلا بالشكل؛ فصيغة مجموع الجداءات متضمنة في صيغة جداء المجموعات، إلا أن الأخيرة تمتاز عن الأولى بأنها أبسط من حيث عدد الوصلات اللازمة للتنفيذ، وكذلك من حيث عدد المداخل لمختلف أنواع البوابات المنطقية وهذا ما توضحه المخططات التالية:



الدارة الممثلة للدالة بشكل مجموع جداءات



الدارة الممثلة للدالة بشكل جداء مجاميع

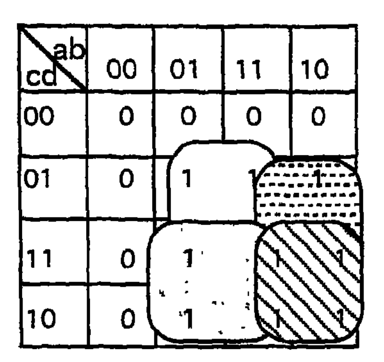
كانت الفائدة هنا من البحث عن الصيغة الصغرى على شكل جداء مجاميع هي التقليل من عدد الوصلات واستخدام بوابات عدد مداخلها أقل؛ إلا أن الأمر يتمثل أحياناً في الحصول على صيغة يتطلب تنفيذها عدداً أقل من البوابات.

مثال : لتكن الدالة المنطقية ذات جدول الحقيقة التالي:

cd \ ab	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

يمكن، بحسب شكل كتابة الدالة، الحصول على صيغتين صغريين

متكافئتين للدالة :



مجموع جداءات

cd \ ab	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

جداء مجاميع

فمن الشكل الأول نجد:

$$F = ac + bc + bd + ad$$

أما من الشكل الثاني فنجد:

$$F = (a + b)(c + d)$$

ومن الواضح أن الصيغتين متكافئتان، إلا أن الأخيرة تحتاج إلى عدد أقل من البوابات ومن الوصلات، إذ تتطلب صيغة مجموع الجداءات أربع بوابات AND كل منها بمدخلين وبوابة OR بأربعة مداخل، أما صيغة جداء المجاميع فتتطلب بوابتين OR كل منها بمدخلين وبوابة AND بمدخلين أيضاً. أما الوصلات فتحتاج الأولى إلى إثنتي عشرة وصلة، والثانية إلى ست وصلات.

3 أوضاع عدم الحدوث

تكوّن الحدود المؤلّفة لدالة منطقية تركيباً بين المتغيِّرات المنطقية يعبر عن وضع للدالة المنطقية تكون فيه محققة أو غير محققة، إلا أن بعض التراكيب لا تأثير لها على قيم الدالة، إذ تكون الدالة غير معرّفة في حالة الحدود المشتركة في تلك التراكيب أو تكون مستحيلة. تكوّن هذه التراكيب ما نسميه بأوضاع عدم الحدوث. تسمح

هذه الأوضاع، عند استخدام جداول كارنو، بالحصول على صيغة صغرى وذلك باستغلالها بحسب الحاجة، وإعطائها القيمة التي نرغب فيها بحيث نحصل على أبسط صيغة.

لنأخذ كمثال على ذلك تصميم دارة كشف الزوجية للأرقام من 0 إلى 9 المرمزة بالترميز الاثناني الطبيعي. دخل الدارة الترميز الطبيعي لهذه الأرقام، وخرجها القيمة 1 كلما كان الدخل زوجياً. نحتاج لذلك إلى أربعة متغيرات منطقية a,b,c,d يتكون منها الرقم abcd.

يتألف جدول الحقيقة لدالة الزوجية هذه من أربعة أعمدة وأربعة أسطر. ونضع في خانات الجدول 1 مقابل كل رقم زوجي وصفراً في غير ذلك. ولكن بعض مربعات الجدول تقابل أعداداً لا تدخل في نظام الكشف مثل العدد 12. نضع مقابل هذه الأعداد الإشارة X للدلالة على أنه يمكن الاستعاضة عنها بواحد أو صفر، بحسب الحاجة، عند البحث عن الصيغة المبسطة للدالة الممثلة لدارة كشف الزوجية. وبذلك نجد الجدول التالي:

cd \ ab	00	01	11	10
00	1	1	X	1
01	0	0	X	0
11	0	0	X	X
10	1	1	X	X

أفضل صيغة مبسطة لدالة الزوجية هي:

$$f = \overline{d}$$

حيث أعطينا لـ X في السطر الأول والسطر الرابع القيمة 1، ولم نكترث بتلك الموجودة في السطرين الثاني والثالث.



الفصل الثالث

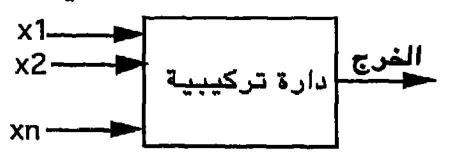
الدارات التركيبية الشهيرة

1 مقدمة

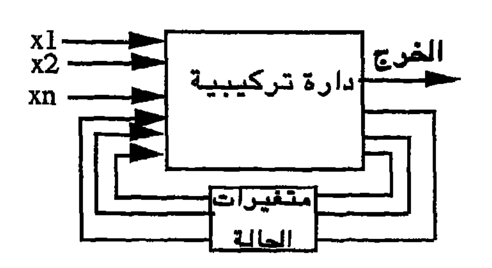
تُستَخدَمُ بعض الدوال المنطقية الخاصة ببعض العمليات بكثرة عند تصميم بعض النظم المنطقية. دُرست هذه الدوال والعمليات وصممت لها دارات منطقية خاصة يمكن استخدامها مباشرة دون اللجوء في كل مرة إلى تحقيقها انطلاقاً من البوابات المنطقية الأساسية، مثل بوابات AND أو OR الخ...

تقسم النظم المنطقية إلى نوعين هما:

- <u>النظم التركيبية</u>، التي لا يتعلق فيها الخرج في لحظة ما إلا بالدخل في اللحظة نفسها؛ ولها التمثيل التالي:



- <u>النظم التتابعية</u>، التي يتعلق فيها الخرج في لحظة ما بالدخل في هذه اللحظة وبماضي الجملة (التذكر)، وهو ما نسميه بمتغيرات الحالة التي تختزن ماضي الجملة. ولهذه النظم التمثيل التالي:



نستعرض فيما يلي بعض الدوال التركيبية وطريقة تحقيقها، دون الدخول في التفاصيل، بالرغم من أهميتها أحياناً، حتى لا نبتعد عن غرضنا من هذا الكتاب. وسنعالج في الفصلين السادس والسابع النظم التتابعية.

2 دارة جمع الأعداد الاثنانية

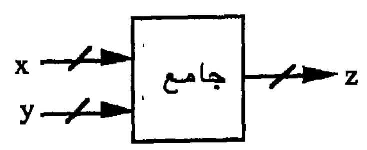
ليكن العددان x و y الممثلان اثنانياً بالشكل التالي:

$x : x_n \, x_{n-1} \, ... \, x_1$

$y : y_n \, y_{n-1} \, ... \, y_1$

وقد وضعنا تمثيل العددين في n خانة آخذين بالاعتبار أن بعض الخانات العليا قد تكون كلها معدومة $(x_n = x_{n-1} = ... = x_k = 0)$ أو $(y_n = y_{n-1} = ... = y_k = 0)$. ولن يغير هذا شيئاً من الهدف الذي نبحث عنه، وهو جمع العددين x و y.

المطلوب إذن تصميم دارة تقوم بهذه العملية ولها المخطط الصندوقي:



(يشير الخط المعترض إلى أن المداخل والمخارج ممثلة في أكثر من خانة اثنانية.)

من الواضح أن ناتج الجمع يجب أن يكون ممثلاً في n +1 خانة، كما

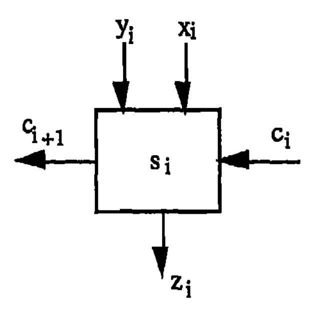
هو الحال عند جمع عددين عشريين.

تجري عملية الجمع الاثناني، كما بينا سابقاً، بطريقة مشابهة لطريقة جمع الأعداد العشرية. لنشر بـ C_i إلى باقي جمع الخانة x_i إلى الخانة y_i. تجري عملية الجمع على النحو التالي:

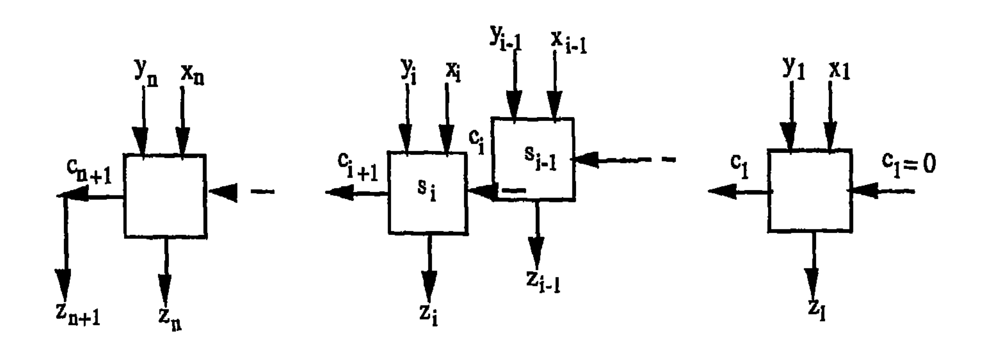
$$\begin{array}{r} c_{n+1}\, c_n\, c_{n-1} \cdots c_1 \\ x_n\, x_{n-1} \cdots x_1 \\ y_n\, y_{n-1} \cdots y_1 \\ \hline z_{n+1}\, z_n\, z_{n-1} \cdots z_1 \end{array}$$

حيث $c1 = 0$ و $zn + 1 = cn + 1$

نلاحظ أن عملية الجمع هي نفسها مهما يكن موقع الخانة، إذ نجمع x_i إلى y_i إلى c_i (حيث c_i هو باقي جمع x_{i-1} إلى y_{i-1}) لنحصل بعد ذلك على z_i والباقي c_{i+1} الذي نستخدمه عند جمع x_{i+1} إلى y_{i+1}. على هذا، فلإيجاد الدارة المنطقية التي تجمع العددين x و y، يكفي أن نصمم دارة جزئية s_i لجمع x_i و y_i و c_i، ثم نصل هذه الدارات بعضها ببعض لنحصل على الدارة الكاملة لجمع العددين x و y. فإذا رمزنا إلى دارة الجمع الجزئية بالمخطط الصندوقي التالي:



وجمعنا العدد اللازم من هذه مكونات جزئية، حصلنا على دارة الجمع الكلية وفق المخطط الصندوقي التالي:



لتصميم دارة الجمع الجزئية، نضع جدول الحقيقة لعملية الجمع الجزئية، حيث نخصص جدولاً لناتج الجمع وجدولاً آخر للباقي:

$c_i \backslash x_i y_i$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

z_i

$c_i \backslash x_i y_i$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

c_{i+1}

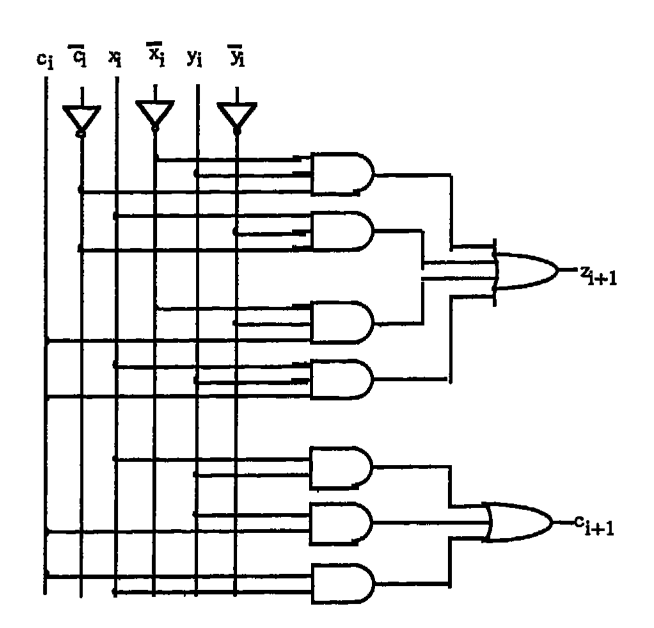
من الجدولين السابقين نجد:

$$z_i = \overline{x}_i\, y_i\, \overline{c}_i + x_i\, \overline{y}_i\, \overline{c}_i + \overline{x}_i\, \overline{y}_i\, c_i + x_i\, y_i\, c_i = x_i \oplus y_i \oplus c_i$$

$$c_i + 1 = x_i\, y_i + c_i\, y_i + c_i\, x_i$$

ويمكن تحقيق هاتين الدالتين باستخدام بوابات منطقية أساسية على نحو ما هو مبين في الشكل التالي. نسمي دارة الجمع الجزئية هذه بالجامع التام full adder. ونلاحظ أنه من الممكن بناء دارة الجامع الكاملة انطلاقاً من دارات جمع جزئية أخرى، مثل الدارة التي تجمع عددين (يتألف كل منهما من خانتين) $x_{i+1}x_i$ و $y_{i+1}y_i$ وباقي الجمع c_i، وتعطي على خرجها ناتج الجمع z_i z_{i+1} والباقي c_{i+1}.

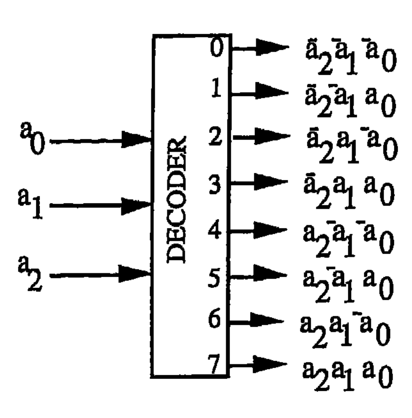
(يمكن بناء هذه الدارة كتمرين.)



3 دارة مفكك الترميز

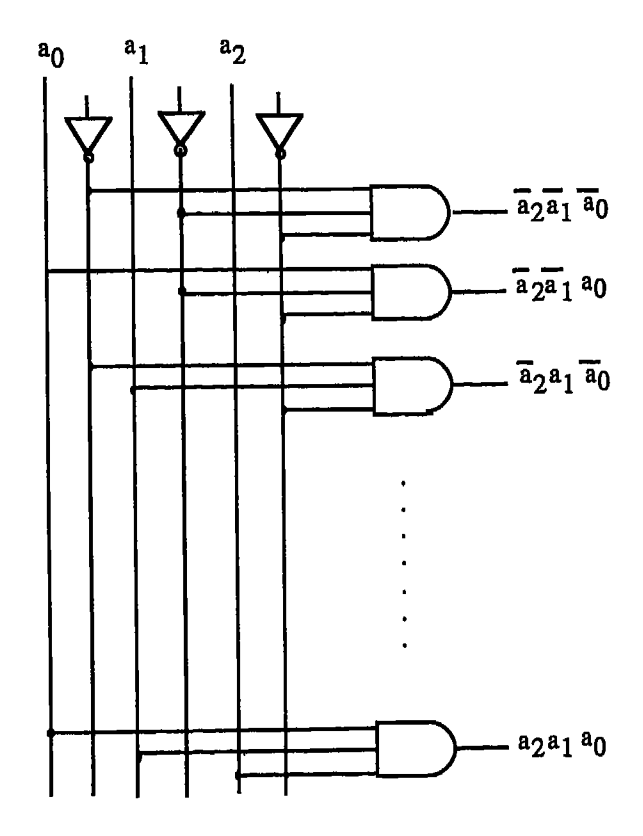
مفكك الترميز Decoder دارة تركيبية متعددة المداخل والمخارج. دخل الدارة عدد ممثل بالترميز الاثناني الطبيعي في n خانة، وخرجها مؤلف من 2^n مخرجاً منفصلاً تكون كلها على القيمة صفر (أو واحد) ما عدا الخط ذا الترتيب المساوي للرقم المسجل على الدخل، الذي يأخذ القيمة واحد (أو صفر).

فإذا كان العدد ممثلاً بثلاث خانات مثلاً، يكون لهذه الدارة ثمانية مخارج تتحدد قيمة كل منها من قيمة الدخل بالشكل التالي:



مفكك الترميز

وتعطى الدارة المنطقية التي تحقق هذه العملية بالمخطط التالي:



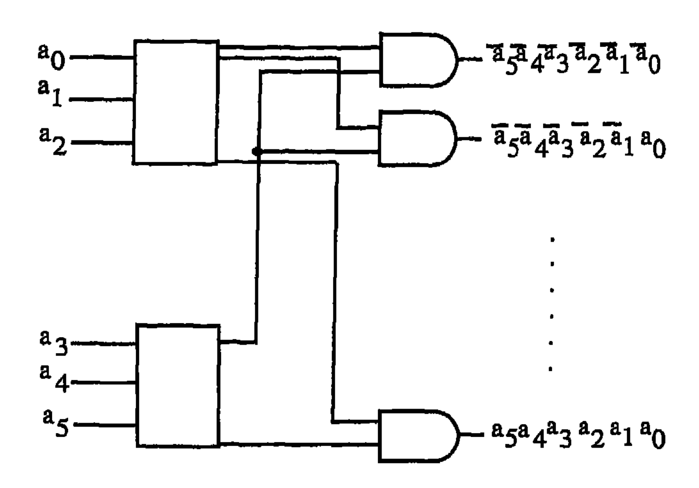
نرى في هذه الدارة أننا محتاجون إلى ثماني بوابات منطقية، لكل بوابة ثلاثة مداخل، لتحقيق مفكك ترميز بثلاثة مداخل. ومن الواضح أنه لايمكن عملياً تحقيق دارة مفكك ترميز وحيدة لأي عدد من المداخل. فإذا أردنا تنفيذ مفكك ترميز بستة عشر مدخلاً، سنحتاج إلى 2^{16} بوابة، لكل بوابة ستة عشر مدخلاً، وهذا عدد كبير جداً يجعل تنفيذ الدارة صعباً تقانياً.

تتوافر مفككات ترميز بثلاثة أو أربعة أو خمسة مداخل على شكل دارات متكاملة، يمكن استخدامها استخداماً منفصلاً، أو تجميع عدد منها لتكوين مفكك ترميز بمداخل أكثر. وفي هذا الخصوص يمكن اتباع القاعدة التالية:

ليكن المطلوب بناء مفكك ذي n خانة، وليكن أكبر مفكك متوافر في شكل دارة متكاملة هو مفكك ذو دخل من m خانة، عندئذ نبحث عن أكبر عدد صحيح r يحقق العلاقة $m \geq \frac{n}{k} \geq r$، حيث k هو عدد صحيح من القوى الصحيحة لـ 2.

مثال:

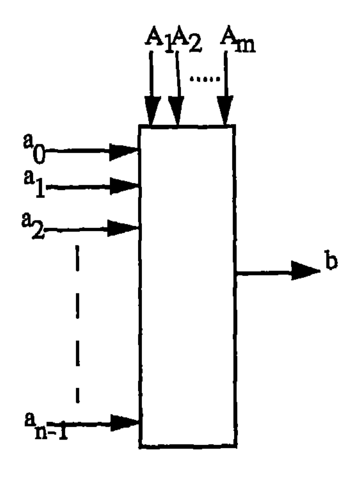
ليكن لدينا مفككات ترميز أكبرها ذو عدد مداخل $m = 5$، وليكن المطلوب هو بناء مفكك ترميز عدد مداخله $n = 6$. عندئذ نجد $5 \geq \frac{6}{k}$، ومنه $k = 2$ ومن ثم $3 = \frac{6}{2} \geq r$. لذا نستخدم اثنين من المفككات ذات الثلاثة مداخل ونربطها على الشكل التالي:



4 دارة الناخب

نحتاج في النظم المنطقية (مجموعة من الحواسيب مثلاً) إلى وصل منظومة بمنظومات أخرى من أجل تبادل المعلومات فيما بينها باستخدام خط اتصال وحيد. ولما كانت المنظومة المنطقية لا تتصل في الآن نفسه بأكثر من منظومة واحدة، كان من الضروري تصميم دارة «انتخاب» بين مجموعة من الخطوط لقراءة خط واحد في لحظة ما.

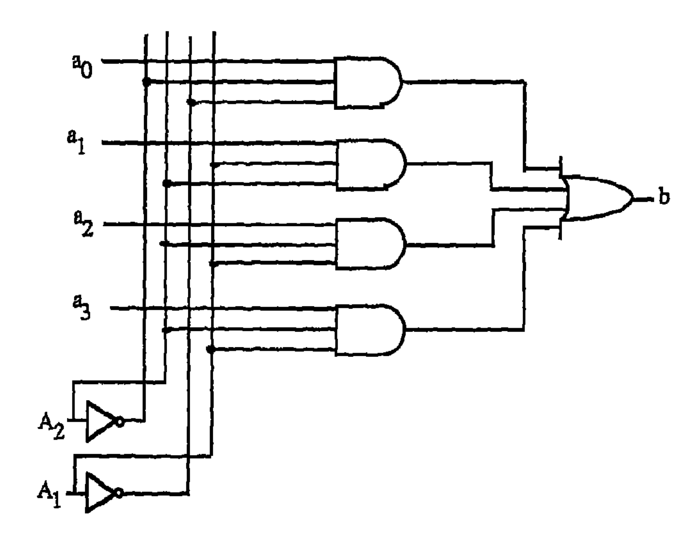
المخطط الصندوقي للناخب Multiplexer هو التالي:



حيث a_i هي خطوط المعلومات، و b هو خرج الناخب. أما المداخل $A_1, A_2 \dots A_m$ فهي مداخل الانتخاب التي تحدد قيمتهاالعددية رقم المدخل الذي سيكون موصولاً إلى الخرج b.

لنلاحظ أن m و n ترتبطان بالعلاقة $2^m \geq n$، فإذا كان لدينا ثمانية مداخل $a_0\, a_1 \dots a_7$ احتجنا إلى ثلاثة مداخل انتخاب $A_1\, A_2\, A_3$ ($2^3 = 8$)، فإذا أردنا قراءة المدخل الرابع a_3 مثلاً، فعلى مداخل الانتخاب أن تأخذ القيمة A1 A2 A3 = 011.

إن بناء دارة الناخب هو عملية مباشرة، ويبين المخطط التالي دارة الناخب بأربع قنوات.



نلاحظ أن دارة الناخب هي دارة بثلاثة مستويات: مستوى العواكس للمداخل A_2, A_1 ثم مستوى بوابات AND، وأخيراً مستوى بوابة OR؛ لذا يمكن التفكير في استخدام هذه الدارة في تحقيق دوال منطقية بسيطة بواسطة دارة متكاملة وحيدة.

لنأخذ المثال البسيط الآتي:

لتكن الدالة ذات جدول الحقيقة التالي:

z \ xy	00	01	11	10
0	1	0	1	0
1	1	1	0	1

والتي تعطى صيغتها المبسطة بالعلاقة التالية:

$$f = \overline{x}\,\overline{y} + z\overline{x} + \overline{z}xy + x\overline{y}z$$

يمكن تحقيق هذه الدالة باستخدام ناخب وحيد بأربعة مداخل، وذلك باجراء التوصيلات التالية:

$x = A_1, y = A_2$

$a_0 = 1, a_1 = z, a_2 = \overline{z}, a_3 = z$

إذن، فلتحقيق هذه الدالة نستخدم دارة ناخب وحيدة مع عاكس.

5 الذاكرة

الذاكرة Memory دارة لحفظ المعلومات بصيغتها الاثنانية. ولها أنواع عدة أهمها:

- ذواكر ROM (Read only Memory): وهي ذواكر تخزن فيها المعلومات مرة وحيدة، ويمكن بعدها قراءتها فقط دون إمكان إعادة الكتابة فيها لاحقاً.
- ذواكر EPROM (Erasable Programmable ROM): وهي ذواكر يمكن الكتابة فيها وقراءة ما كتب فيها، وتبقى المعلومات مخزونة في الذاكرة ولو انقطع التيار الكهربائي. تختلف الـ EPROM عن الـ ROM في قابليتها لمسح محتوياتها وإعادة الكتابة فيها من جديد.
- ذواكر RAM (Random Access Memory): وهي ذواكر يمكن الكتابة فيها وقراءتها في كل لحظة، ولكنها تفقد المعلومات المدونة فيها عند انقطاع التيار الكهربائي.

تتميز الذواكر بالمواصفات الرئيسية التالية:

- عرض الكلمة: ونقصد بذلك عدد الخانات الاثنانية المكوّنة

للكلمة الواحدة، فمنها ذواكر بعرض كلمة 4 خانات اثنانية أو 8 خانات اثنانية...

- سعة الذاكرة: ويقصد بذلك عدد الكلمات التي يمكن تخزينها في الذاكرة، مثلاً 2 كيلو كلمة (2KW)، 4 كيلو كلمة... والكيلو كلمة تساوي إلى 2^{10} كلمة، أي 1024 كلمة.
- سرعة القراءة: ويقصد بذلك الزمن اللازم للوصول إلى كلمة معينة، اعتباراً من اللحظة التي يعطى الطلب للذاكرة وحتى لحظة جاهزية الكلمة للقراءة.

بنية الذاكرة ROM

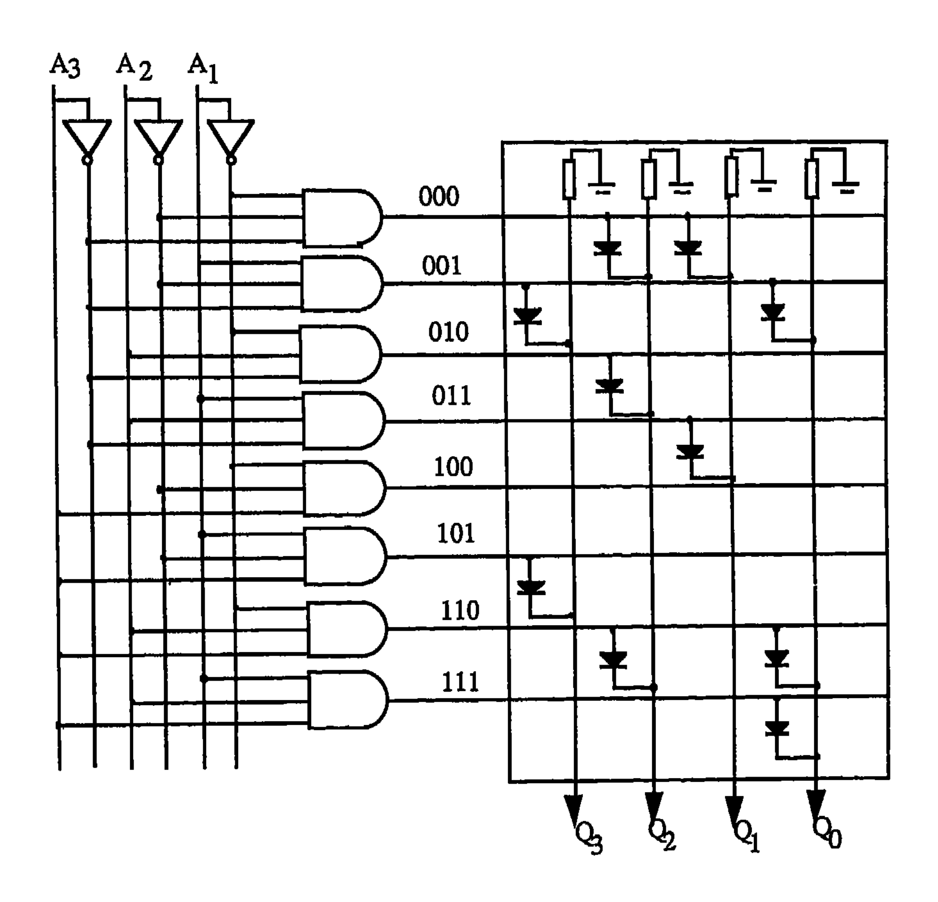
تتألف الذاكرة من جزأين رئيسيين، أولهما مفكك ترميز العنوان، المؤلف عادة من n خط دخل والذي يسمح بعنونة 2^n كلمة. (إذا كان عرض الكلمة هو m خانة اثنانية فيمكن تخزين $m.2^n$ خانة اثنانية في الذاكرة). أما الجزء الثاني فهو عادة مصفوفة ديودات أو ترانزستورات لتكوين الكلمات التي تحتويها الذاكرة. يقابل كل ديود أو ترانزيستور خانة في كلمة، ويقابل كل سطر من الديودات أو الترانزستورات كلمة. ويقابل عدد الأسطر عدد الكلمات التي يمكن للذاكرة تخزينها.

تستخدم هذه الذاكرة لحفظ معلومات نحتاج إليها في عمل المنظومة المنطقية التي تشكل الذاكرة جزءاً منها. كما تستخدم في تحويل التراميز أو توليد الكلمات أو في توليد الدوال تامنطقية...

سنبين فيما يلي بنية ذاكرة ROM بمثال توضيحي لذاكرة سعتها ثماني كلمات وعرض كلماتها أربع خانات اثنانية. لنفترض أن هذه الذاكرة تضم الكلمات التالية:

0110
1001
0100
0010
0100
0000
1000
0101
0001

أي إننا نريد أن نرى على العنوان 0000 القيمة 0110 وعلى العنوان 0001 القيمة 1001 وهلم جرا... ويتحقق ذلك باستخدام مفكك ترميز بثلاثة مداخل ومصفوفة ديودات مثلاً، كما في الشكل التالي:



فعندما يكون العنوان الموجود على الدخل هو 000 يصل جهد موجب إلى مصعد الديودات الموجودة في السطر الأول، في حين تكون مصاعد الديودات في الأسطر الأخرى على جهد سالب أو معدوم، من ثم سنجد على المخرج Q3Q2Q1Q0 الجهود $V^-V^+V^+V^-$ التي تكافيء القيمة 0110.
وعندما يكون العنوان هو 101 سيكون الجهد على هذا السطر من مصفوفة الديودات موجباً في حين يكون الجهد سالباً أو معدوماً على الأسطر الأخرى؛ على هذا نجد على خرج الذاكرة جهوداً تقابل القيمة الاثنانية 1000.

تنتج هذه الذواكر في شركات تصنيع الدارات المتكاملة التي يُرسل إليها جدول القيم الاثنانية المراد تخزينها في الذاكرة.

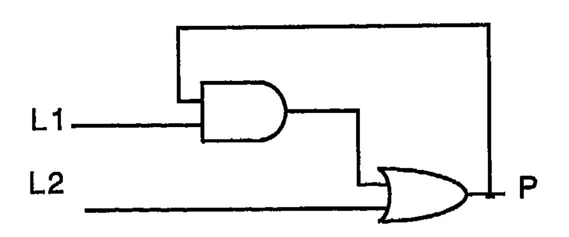


الفصل الرابع

الدارات التتابعية الشهيرة

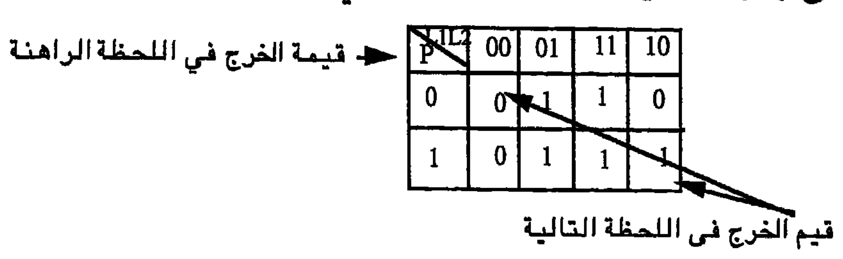
1 مقدمة

تميزت الأنظمة والدارات التي عالجناها حتى الآن بكون خرجها في لحظة ما تابعاً لدخلها في اللحظة نفسها فقط. وقد سميناً هذه النظم والدارات بالنظم التركيبية، نظراً لكون الخرج هو مجموع تراكيب من إشارات الدخل. ولكن بعض الجمل لا يمكن معرفة خرجها من مجرد معرفة دخلها في لحظة ما، وإنما نحتاج إلى معرفة دخلها في الماضي، أو ما نسميه بحالة الجملة أو متغيراتها الداخلية التي تختزن ماضيها وتمثّل ذاكرتها. لنأخذ مثالاً الدارة المبينة في المخطط التالي:



فهل يكفي لمعرفة خرج هذه الدارة في لحظة ما معرفة دخلها في تلك اللحظة؟ الجواب بالنفي، ذلك أن الخرج هو في الوقت نفسه دخل للدارة أيضاً، ويجب معرفة قيمته عند تطبيق المداخل L1, L2 كي نتمكن من معرفة قيمة الخرج.

يعطى جدول الحقيقة لهذه الدارة كما يلي:

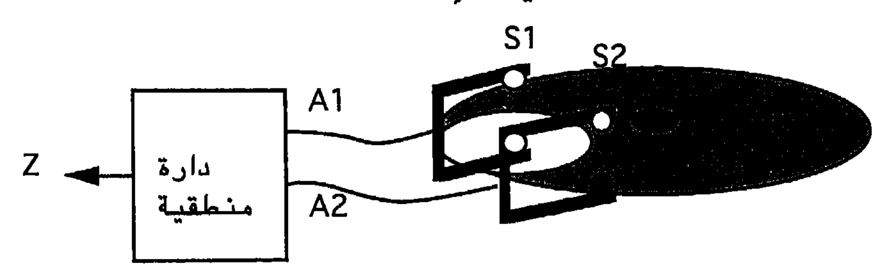


P \ L1L2	00	01	11	10
0	0	1	1	0
1	0	1	1	1

يربط هذا الجدول قيمة الدخل والخرج في اللحظة الراهنة، كما يعطي قيمة الخرج في اللحظة التالية (داخل خانات الجدول)، أي بعد زمن يساوي زمن الانتشار في البوابات المكوّنة للدارة.

<u>مثال</u>: مسألة كشف الاتجاه:

لتكن الجملة المؤلفة من قرص معدني مفتوح في إحد أجزائه (كما في الشكل التالي)، ومن جملتين من المُحسّات S1 و S2 يتكون كل منهما من مرسل ومستقبل ضوئيين. يقوم القرص أثناء دورانه بحجب الأشعة الضوئية بين المرسل والمستقبل أثناء وجود الجزء المعتم من القرص بينهما، ويكون الأمر عكس ذلك عند وجود الجزء المفتوح من القرص بين طرفي المُحِسّ.



يعطي كلا المحسين إشارة واحد منطقي عندما يكون الضوء محجوباً بين المرسل والمستقبل (Ai =1)، وإشارة صفر منطقي في الحالة المعاكسة (Ai =0). والمطلوب كشف اتجاه حركة دوران القرص وإعطاء إشارة على خرج الدارة على النحو التالي: واحد عندما تكون جهة الدوران موجبة (بعكس عقارب الساعة) وصفر عندما تكون جهة

الدوران سالبة.

لنلاحظ أن كشف جهة الدوران يحدث بناء على المعلومات التي تأتي من المحسات الضوئية، ومن ترتيب التتالي في القيم التي تظهر على خرجي المحسين؛ فإذا كان A1A2=10 أولاً ثم أصبح A1A2=11 فهذا يعني أن جهة الدوران موجبة؛ أما إذا كان A1A2=01 أولاً ثم أصبح A1A2=11 فهذا يعني أن جهة الدوران سالبة. وبالمشابهة، في حال كان A1A2=10 ثم أصبح A1A2=00 فهذا يعني أن جهة الدوران أصبحت سالبة، الخ...

نجد مما تقدم أن علينا، عند أي تغير في قيم مخارج المحسات، «تذكّر» قيمة المخارج السابقة. ومن القيم السابقة والقيم الحالية (المغايرة للقيم السابقة) يمكن أن نحدد جهة الدوران.

مثال: مسألة إرسال معلومات على خط اتصال وحيد

تأخذ المعلومات شكل تتالٍ محدد من القيم المنطقية، يكوّن كلمة مؤلفة من ثمان خانات اثنانية (كالكلمة10010111). ترسل الخانات الواحدة تلو الأخرى، من اليسار إلى اليمين. لِتَعَرُّف الكلمة، يجب عند استقبال الخانة الثامنة تذكر الخانات السبع السابقة التي جرى استقبالها في لحظات سابقة. وتجري هذه العمليات عادة عن طريق «تخزين» تلك القيم كما سنرى لاحقاً.

نسمي هذا النوع من الجمل المنطقية بالجمل التتابعية، إذ تعتمد في أدائها، كما يشير اسمها، على تذكر التتابع في مداخلها. وتنقسم الجمل التتابعية إلى نوعين: الجمل التتابعية اللامتزامنة، والجمل التتابعية المتزامنة.

• في الجمل المتزامنة، يكون التغير في الجملة مرهوناً بأمر إضافي خارجي متغير بين مستويين: الصفر والواحد، نسميه «الساعة»، وغالباً ما يكون هذا التغير دورياً.



عمل الساعة مشابه لعمل ضابط الإيقاع في فرقة موسيقية، إذ لا يحدث الانتقال من وضع إلى آخر إلا في لحظات محددة، تؤخذ عادة عند تغير قيمة الإشارة الدورية من 1 إلى 0 (جبهة صاعدة) أو عند تغيرها من 0 إلى 1 (جبهة هابطة)، أو ما دامت أن قيمة الإشارة الدورية مساوية لـ 1 أو 0. وفي جميع الأحوال، لا تقوم الجملة إلا بانتقال وحيد خلال دور الساعة، ولو تغيرت المداخل الخارجية بعد هذا الانتقال خلال دور الساعة.

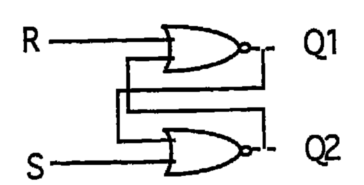
• في الجمل اللامتزامنة، يكون الانتقال من وضع إلى آخر مرهوناً بالمتغيرات الخارجية والمتغيرات الداخلية، وذلك بصرف النظر عن لحظة حدوث التغير.

مسألة كشف الاتجاه المذكورة آنفاً مثال على جملة تتابعية لامتزامنة. أما مسألة إرسال المعلومات تسلسلياً فهي مثال على جملة تتابعية متزامنة، ذلك أننا لن نتمكن من تعرف تتالٍ محدد من القيم المنطقية بدون استخدام إشارة إضافية تحدد لنا لحظة التمييز بين خانتين اثنانيتين متماثلتين!

2 دارة القلاب RS

لندرس الآن دارة شهيرة هي دارة القلاب RS، بطرازيها المتزامن واللامتزامن.

لتكن الدارة المبينة في الشكل التالي والمؤلفة من بوابتين NOR رُبط خرج كل منهما بدخل الأخرى (المتغيرات الداخلية) وطبقت عليهما المداخل الخارجية R و S.



لخرج هذه الدارة قيم متمايزة، تبعاً للقيم المطبقة على دخلها ولقيم المخارج في لحظة تطبيق الدخل. وإذا استثنينا الدخل RS =11، الذي قد يُدخل الدارة في حالة اهتزاز، تكون قيم الخرج متعاكسة دائماً أي $Q_1 = \overline{Q}_2$.

في هذه الحالة يمكن أن نحصر اهتمامنا فقط بـ Q1 (الذي سنشير إليه من الآن فصاعداً بـQ). بناء على ما تقدم، يمكن اعتبار أوضاع العمود المقابل للدخل الخارجي 11 كأوضاع عدم حدوث، مما يؤدي إلى الجدول التالي:

Q \ RS	00	01	11	10
0	0	1	X	0
1	1	1	X	0

لنحاول الآن قراءة هذا الجدول. في العمود ذي الدخل الخارجي 00، نلاحظ أن المتغيرات الداخلية تحافظ على قيمتها مابين لحظة واللحظة التالية. أما في حالة الدخل الخارجي 01، فتأخذ المتغيرات الداخلية القيمة 1 مهما تكن قيمة المتغيرات الداخلية في اللحظة السابقة. وأخيراً يؤدي الدخل الخارجي 10 بالمتغيرات الداخلية إلى أخذ القيمة 0 مهما تكن قيمة هذه المتغيرات في لحظة سابقة.

يمكن إذن إسناد دور الذاكــرة إلى هذه الدارة التي تعطي على خرجها Q القيمة 0 من أجل الدخل الخارجي RS = 10 ، والقيمة 1 من أجل الدخل الخارجي RS = 01. وتحافظ الدارة على القيمة المسجلة فيها (1 أو 0) عند تطبيق الدخل الخارجي RS = 00.

وخلاصة ما سبق: يضع الدخل RS = 10 خرج الدارة على القيمة 0 (عملية Reset)، كما يضع الدخل RS = 01 خرج الدارة على القيمة 1

(عملية Set)، أما الدخل RS = 00 فيسمح بحفظ قيمة خرج الدارة.

لنلاحظ أن استخدام هذه الدارة كذاكرة يجري على مرحلتين: الأولى تسجيل القيمة التي نرغب فيها (RS = 01 في حالة القيمة 1 و RS = 10 في حالة القيمة 0)، والثانية حفظ هذه القيمة بتطبيق الدخل RS = 00 حتى إشعار آخر.

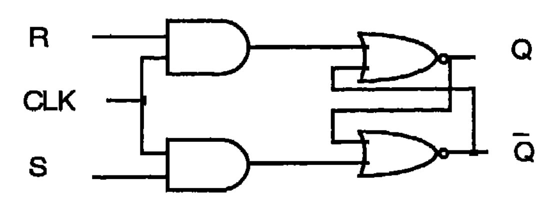
بالعودة إلى الجدول السابق، نجد المعادلة الواصفة لهذه الدارة:

$q = S + \overline{R}\ Q$

حيث تشير q إلى قيمة الخرج بعد تطبيق الدخل عليها واستقرارها، أما Q فتشير إلى قيمة الخرج لحظة تطبيق الدخل عليها؛ أي إن q هي قيمة الخرج في اللحظة التالية. وتسمى هذه الدارة بدارة القلاب RS (RS Flip-Flop).

هذه الدارة هي دارة تتابعية، ذلك أنه لا يمكن معرفة خرجها من مجرد معرفة الدخل الخارجي، إذ لابد أيضاً من معرفة قيمة المتغير الداخلي Q لتحديد خرج الدارة. وهي تتابعية لا متزامنة، ذلك أن عملها لايخضع لأي إيقاع خارجي، إنما يرتبط مباشرة ولحظياً بقيمة الدخل الخارجي وبقيمة المتغير الداخلي.

لتحويل هذه الدارة إلى دارة متزامنة، يمكن ربط مدخليها الخارجيين R و S بإشارة متغيرة بين الواحد والصفر (ساعة) عبر بوابتي AND على النحو التالي:



تكون الدارة في حالة حفظ أو تخزين مادامت قيمة الإشارة الدورية هي الصفر (CLK = 0)، وعندما تصبح إشارة الساعة مساوية الواحد، يأخذ خرج الدارة القيمة التي نشاء تبعاً لقيم R و S. ولكن ماتقدم لا يكفي لجعل الدارة السابقة متزامنة بمقتضى التعريف الذي

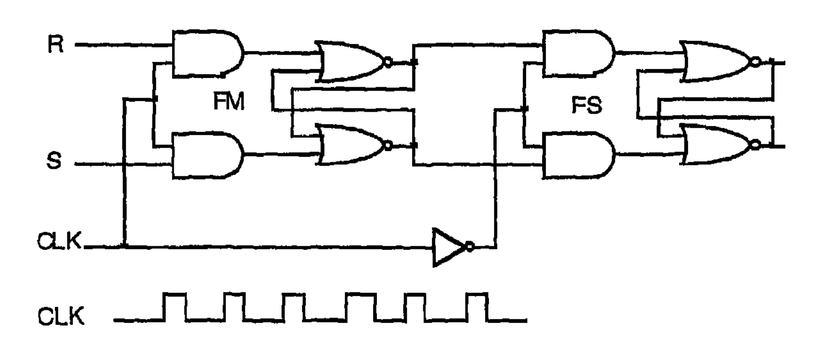
قدمناه سابقاً، والذي يفرض ألا تقوم الدارة المتزامنة بأكثر من تغير واحد خلال دور واحد من أدوار الساعة، إذ نلاحظ هنا أنه خلال بقاء الساعة على الواحد، يمكن أن يؤثر المداخلان R و S في خرج الدارة أكثر من مرة، خاصة إذا كان بقاء الساعة على الواحد يدوم لفترة أطول بعدد من المرات من زمن الانتشار في البوابات المستخدمة، لذا يجب إيجاد حل لهذه المسألة بحيث تصبح الدارة السابقة متزامنة! (هذه المسألة هامة جداً لعمل الدارات المتزامنة التي يرتبط بعضها ببعض لتستجيب لوظيفة محددة.)

يمكن توفير التزامن عن طريق تحديد مدة بقاء الساعة على القيمة 1 تحديداً يسمح بتغير وحيد في الدارة، إلا أن لهذه الطريقة مخاطر كبيرة. سنعرض فيما يلي طريقتين لمعالجة مشكلة التزامن:

- طريقة السيد والتابع Master-Slave.
- طريقة التزامن بالجبهة الصاعدة أو بالجبهة الهابطة.

1-2 القلاب المتزامن بطريقة السيد والتابع

نستخدم في هذه الحالة قلابين متزامنين من نوع RS يوصل خرج الأول (السيد) بدخل الثاني (التابع)، ونصل الساعة CLK إلى مدخل التزامن للأول، في حين نصل إلى مدخل التزامن في الثاني مقلوبَ إشارة الساعة.



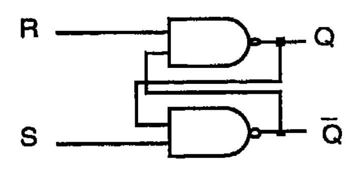
وعلى هذا يكون القلاب الأول FM في وضع قابل للتأثر بما يوضع على

مدخليه R و S عندما تكون قيمة الساعة مساوية للواحد، ويبقى هذا القلاب محافظاً على آخر خرج له عند تغير الساعة من الواحد إلى الصفر. عندها يبدأ القلاب الثاني FS بالتأثر بما سُجّل في القلاب الأول، ويحافظ على قيمة خرجه حتى الصعود التالي لإشارة الساعة إلى القيمة واحد، بدون أن يتأثر بما قد يحدث من تغيرات على مداخل القلاب الأول خلال هذه المدة، وبصرف النظر عن مدة بقاء الساعة على القيمة واحد (حالة النبضة الرابعة مثلاً في الشكل السابق). ومن ثم، إذا نظرنا إلى القلابين كقلاب وحيد، مدخلاه هما مدخلا الأول ومخرجاه هما مخرجا الثاني، فعندئذ يحقق القلاب الجديد تعريف الدارة المتزامنة من حيث إنها لا تغير من وضعها (خرجها) إلا مرة واحدة خلال تغير الساعة من الصفر ثم إلى الواحد ثم إلى الصفر.

2-2 القلاب المتزامن بالجبهة الصاعدة أو بالجبهة الهابطة

في نمط التزامن بالجبهة، يحدث التغير في الدارة المتزامنة عند تغير الساعة من القيمة 0 إلى القيمة 1 (ومنه التسمية «الجبهة الصاعدة») أو عند تغير الساعة من القيمة 1 إلى القيمة 0 (ومنه التسمية «الجبهة الهابطة»).

قبل أن نخوض في دارة القلاب RS المتزامن، سنقوم ببناء دارة جديدة تقوم بالوظيفة نفسها، ولكن باستخدام بوابات من نوع NAND:



يمكن أن ندرس مختلف أوضاع هذه الدارة كما فعلنا في حالة الدارة المبنية بواسطة بوابات NOR، حيث يؤدي المدخلان R و S

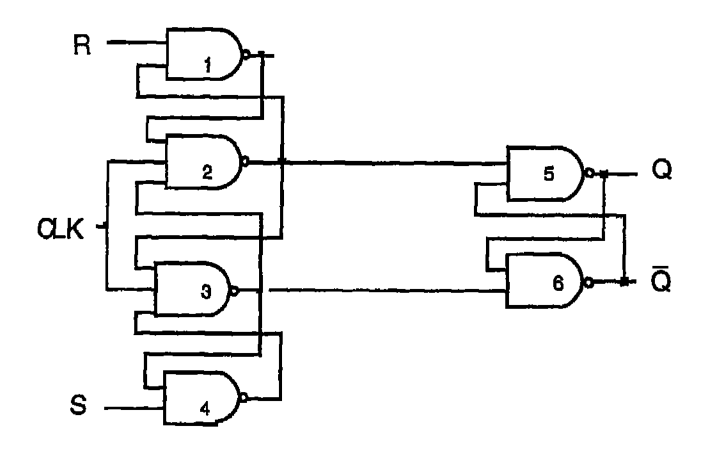
الدور السابق نفسه، في حين يؤدي وضع المداخل RS = 11 إلى المحافظة على القيمة السابقة للخرج، أما القيمة RS = 00 فتؤدي إلى الإخلال بعمل دارة القلاب.

للقلاب في هذه الحالة المعادلة التالية:

$$q = \overline{R} + SQ$$

لنعد الآن إلى موضوعنا المتعلق بدارة القلاب RS المتزامن، الذي يعمل على الجبهة الصاعدة. هنا تبقى الدارة منيعة على التغيرات في الدخل، إلا عند تغير إشارة الساعة من الواحد إلى الصفر، ويسمح هذا عندئذ بتغير وحيد.

لنتأمل الآن الدارة المبينة فيما يلي، والمؤلفة من جزأين:



البوابات 1 و 2 و 3 و 4 في الدخل، وترتبط مباشرة بالمداخل الخارجية R وS؛ وبوابات الخرج 5 و 6 التي تكوّن قلاباً RS. ولندرس أداء هذه الدارة عند تطبيق قيم الدخل المسموحة.

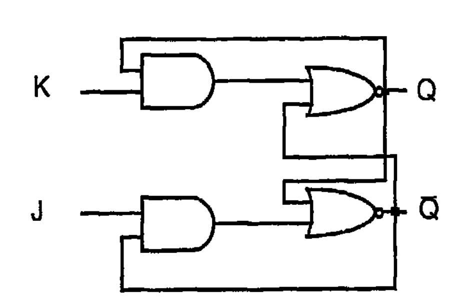
- لنفترض أولاً أن قيمة الدخل هي RS = 01، وذلك خلال وجود الساعة على القيمة صفر (مدة إطول من زمن الانتشار في البوابات). هنا يكون خرج البوابة 1 مساوياً للواحد، أما خرج البوابة 4 فيكون مساويا للصفر. وعند تغير الساعة من الصفر إلى الواحد يكون خرج البوابة 2 مساوياً للصفر في حين يكون خرج البوابة 3 مساوياً

للواحد، ونجد Q=1. لنلاحظ أن مخرجي البوابتين 2 و 3 لن يتغيرا مهما يكن التغير في المدخلين الخارجيين R و S أثناء بقاء الساعة على القيمة واحد، ذلك أن خرج البوابة 2 يؤثر بقيمته الصفر في كلتا البوابتين 1 و 3. أما عند عودة الساعة إلى القيمة صفر، يكون مخرجا البوابتين 2 و 3 مساويين للواحد. ومن ثم تحافظ دارة الـ RS الموجودة في الخرج على Q=1.

- في حالة الدخل RS = 10 تكون المسألة متناظرة تماماً بالنسبة إلى الحالة السابقة، ويكون خرج البوابة 2 مساوياً للواحد في حين يكون خرج البوابة 3 مساوياً للصفر، وذلك عند تغير الساعة من القيمة 1 إلى القيمة 0. لهذا نجد في الخرج القيمة Q=0 التي تحافظ على نفسها ولو تغيرت قيم R و S خلال وجود الساعة على الواحد، ذلك لأن البوابتين 2 و 3 لن تغيرا من قيم مخرجيهما حتى تعود الساعة إلى القيمة صفر، وتأخذ هاتان البوابتان القيمة 1. ومن ثم تحافظ دارة الـ RS الموجودة في الخرج على قيمتها السابقة.
- أخيراً، في حالة كون الدخل RS = 11، قبل تغير الساعة من الصفر إلى الواحد، يكون مخرجا البوابتين 1 و 4 مساويين للصفر، ومن ثم يبقى مخرجا البوابتين 2 و 3 مساويين للواحد، ويبقيان على هذه القيمة مهما يكن التغير اللاحق في الساعة.

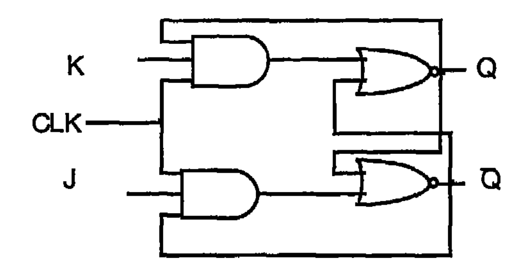
3 دارة القلاب JK المتزامن

إن المشكلة التي واجهتنا في القلاب RS هي كون بعض المداخل ممنوعة (RS = 11 في حالةالقلاب المبني باستخدام بوابات NOR، أو RS = 00 في حالة القلاب المبني باستخدام بوابات NAND) إذ تدخل الدارة في أوضاع تجعل استخدام القلاب محفوفاً ببعض المحاذير، لذا نلجأ عادة إلى تشكيلة أخرى مؤلفة من دارة RS، ولكن بمداخل هي جداء مداخل خارجية في مخارج الـ RS، كما في الشكل التالي:



بفرْض أن خرجي الـ RS متعاكسان، تكون مخارج بوابات الـ AND متعاكسة مهما كانت قيم المدخلين J و K. ومن ثم يكون مدخلا الـ RS متعاكسين ولن يساويا 11 أبداً. أما إذا كان مدخلا JK الخارجيان مساويين لـ 00 فتنتقل هذه القيمة إلى مدخلي الـ RS. لنلاحظ أن إعطاء القيمة 11 للمدخلين الخارجيين يؤدي بالقلاب إلى عكس مخرجيه باستمرار. لذا فعند إضافة مدخل ساعة إلى هذه الدارة وتحويلها إلى دارة متزامنة، تغير الدارة مخارجها عند كل نبضة ساعة.

يعطى المخطط المبدئي لدارة القلاب JK المتزامن بالشكل التالي:



ولهذه الدارة جدول الحقيقة التالي:

Q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

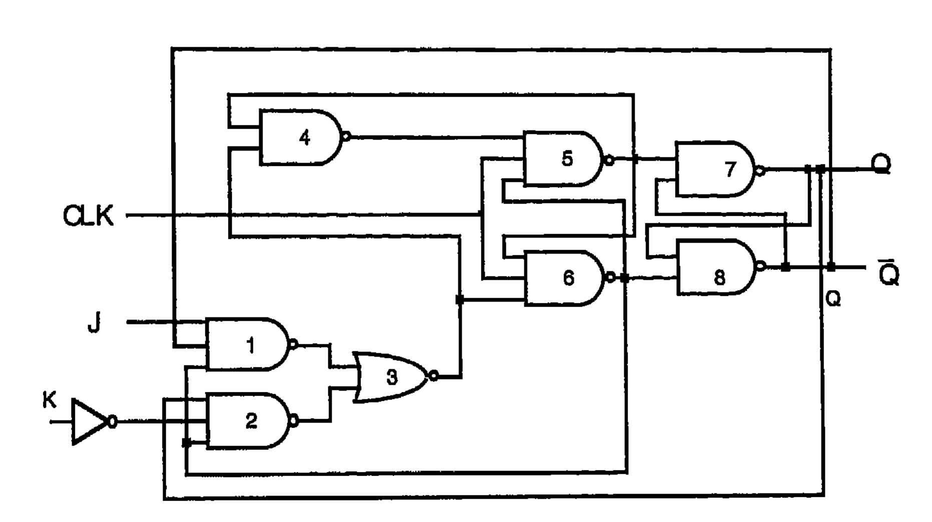
حيث تشير القيم داخل الجدول إلى قيمة الخرج بعد ورود إشارة

الساعة (وهي التي نشير إليها بالرمز q). وبالعودة إلى الجدول السابق نجد معادلة الـ JK الآتية:

$$q = \overline{Q}J + \overline{K}Q$$

لنلاحظ أن التزامن بالشكل الذي عرفناه من قبل لن تحققه دارة الـ JK المبينة آنفاً. يمكن تحقيق التزامن باستخدام تقنية السيد والتابع (كما في دارة الـ RS التي عالجناها سابقاً)، أو بجعل التزامن مرتبطاً بتغير إشارة الساعة من الصفر إلى الواحد (جبهة صاعدة) أو عند تغير إشارة الساعة من الواحد إلى الصفر (جبهة هابطة) . وفيما يلي مخطط لدارة قلاب JK متزامن على الجبهة الصاعدة.

<u>تمرين</u>: يطلب التحقق من أداء هذا القلاب على وجه يطابق جدول الحقيقة المعروض آنفاً.

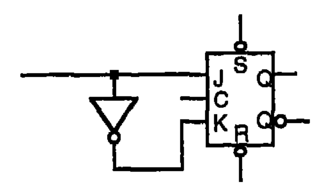


لنلاحظ أخيراً أن المدخل J يقوم بوضع المخرج Q على القيمة 1 عندما يكون JK =10 (أي يقوم بعملية Set) أما المدخل K فيقوم بوضع المخرج Q على القيمة 0 عندما يكون JK = 01 (أي يقوم بعملية Reset). أما في حال إعطاء الدخل القيمة JK = 00 فتحافظ الدارة على القيمة المخزنة فيها سابقاً كما في دارة الـ RS.

<u>تمرين</u>: ماذا يحدث في حال إعطاء الدخل القيمة $JK = 11$؟

4 دارة القلاب D المتزامن

انطلاقاً من دارة القلاب JK يمكن أن نستنتج قلاباً ذا تطبيقات متعددة واستخدام بسيط، إذ يقوم هذا القلاب الجديد بإعطاء قيمة الدخل لحظة تغير الساعة على خرجه، ويحافظ عليها حتى نبضة الساعة القادمة (ولو تغيرت هذه القيمة خلال دور الساعة) ليُظهر من جديد القيمة الموجودة على الدخل طوال الدور القادم. نبني هذا القلاب انطلاقاً من دارة الـ JK بعد وصل المدخل K بالمدخل J عن طريق عاكس، كما في الشكل التالي:



ونحصل على ما نسميه عادة بالقلاب D.

يمكن الحصول على معادلة القلاب D من معادلة القلاب JK بتعويض K بـ $\overline{J}$:

$q = J = D$

تقرأ هذه المعادلة على الشكل التالي: قيمة خرج القلاب D، بعد تأهيل الساعة، هي قيمة الدخل نفسها قبل تأهيل الساعة.

المدخل المشار إليه بـ C في الدارة المبنية آنفاً هو مدخل الساعة. أما المدخلان S (Set) و R (Reset) فهما بالترتيب لوضع المخرج على الواحد أو الصفر في بداية عمل الدارة.

5 استخدام القلابات في العمليات التتابعية

تستخدم القلابات كما رأينا كعنصر ذاكرة، ولكن لخانة واحدة. ويمكن ربط مجموعة من القلابات لاستخدامها كذاكرة لكلمة. كما يمكن استخدام القلابات في تصميم عدادات نحتاج إليها في الحواسيب والنظم المنطقية.

1-5 العدادات

العدادات Counters هي أنظمة تتابعية ذات وظيفة محددة، ينتقل خرجها من قيمة إلى القيمة التالية عددياً عند حدوث تغير معلوم في الدخل. والعدادات عادة دارات ذات عمل حَلَقي، بمعنى أن الخرج الحالي للعداد يتكرربعد دورة عد كاملة.

هناك نوعان أساسيان من العدادات:

- العدادات المتزامنة: وهي عدادات تنتقل من وضع إلى الوضع التالي عند كل نبضة ساعة (إذا كانت المتغيرات الخارجية الأخرى تسمح بذلك)، علماً بأن الساعة هي دخل لجميع قلابات الدارة. ينتمي هذا النوع من العدادات إذن إلى فئة النظم التتابعية المتزامنة.
- العدادات اللامتزامنة: وهي عدادات تنتقل من وضع إلى الوضع التالي بحسب المتغيرات الخارجية.

ثمة نوع آخر من العدادات نسميها بالعدادات المتزامنة/اللامتزامنة، يرتبط جزء منها بالساعة، ويرتبط الجزء الثاني بالجزء الأول ولا يرتبط مباشرة بالساعة.

سنعالج في هذه الفقرة العدادات المتزامنة ذات المدخل الخارجي الوحيد (وهو الساعة نفسها)، وذلك بسبب الخصوصية التي تتمتع بها هذه الأنظمة من حيث التصميم.

يقوم العداد هنا بتسجيل عدد النبضات التي يكون مصدرها الساعة الخارجية (أو أي مصدر آخر يعطي تغيراً بين الصفر والواحد،

مثل محس يتغير خرجه مع مرور صناديق أمامه مثلاً)، ونسمي العداد في هذه الحالة بعداد الحوادث Events Counter.

نعرِّف سعة العداد بكونها مساوية لعدد النبضات التي يمكن للعداد تسجيلها، والتي يمكن قراءتها وفق ترميز العد الاثناني المختار. إن أكثر التراميز شيوعاً هو الترميز الطبيعي.

نستخدم القلابات في تصميم العدادات، فعداد ذو n قلاب له سعة مساوية لـ 2^n. ويؤدي كون عدد الأوضاع المطلوبة N أصغر من 2^n إلى وجود أوضاع ناقصة، ونسمي العداد عندها بعداد ذي ثقوب.

يكون العداد قابلاً للعكس إذا أمكن اختيار جهة العد بين عد تنازلي Down وعد تصاعدي Up.

2-5 تصميم العدادات

يتكون العداد من:

- جملة منطقية تتابعية تستخدم عدداً من القلابات يتوقف عددها على عدد النبضات المراد عدها.
- مجموعة توصيلات راجعة بين القلابات بحيث تكون وضعية القلابات بعد كل نبضة ساعة مطابقة للترميز المطلوب.

ويساوي خرج العداد قيمة متحولات الحالة عند كل نبضة ساعة.

توجد طريقتان لتصميم العدادات. تسمى الأولى بالطريقة المباشرة وتسمى الثانية بطريقة دوال التبديل. سنستعرض فيما يلي طريقة التصميم المباشر فقط، وذلك لسهولة استخدامها وعموميتها حيث يمكن استخدام أي نوع من القلابات.

تُصممّ العدادات وفق الطريقة المباشرة باتباع الخطوات التالية:

1 يحدد نوع القلابات المراد استخدامها في تصميم العداد المطلوب.

2 يحدد في جدول التتالي الذي نريد مشاهدته على الخرج.

3 يحدد في جدول الوضع القادم للعداد بدلالة الوضع الحالي.
4 انطلاقاً من الجدولين السابقين، تُحدَّد قيمة مداخل القلابات، وينظم ذلك في جدول ثالث.
5 نصوغ من جدول كارنو الذي يربط مخارج العداد في اللحظة الراهنة بمخارجه في اللحظة التالية معادلات مداخل القلابات.
6 ننفّذ أخيراً المخطط التقاني للعداد.

مثال: تصميم العدادات باستخدام قلابات JK

ليكن المطلوب تنفيذ عداد، مخارجه A, B, C هي الترميز الطبيعي للأعداد من 0 إلى 7، وذلك باستخدام قلابات من نوع JK.

نرتب في جدول، مؤلف من قسمين، أوضاع العداد في اللحظة الراهنة وأوضاعه القادمة (عند ورود نبضة جديدة):

	A	B	C	a	b	c
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	1	1	1
7	1	1	1	0	0	0

حيث تشير الحروف الصغيرة a, b, c إلى قيمة الخرج في اللحظة القادمة، في حين تشير الحروف الكبيرة A, B, C إلى الخرج في اللحظة الحالية.

نحتاج في هذه المسألة إلى ثلاثة قلابات. ونكتب جدول الحقيقة للقلاب JK الذي يربط الخرج في اللحظة الحالية Q بالخرج في اللحظة القادمة بحسب قيم المداخل J و K:

Q	q	J	K
0	0	0	X
0	1	1	X
1	1	X	0
1	0	X	1

ومن معرفة قيم Q والمداخل J و K التي تؤدي إلى القيم المطلوبة q عند ورود نبضة جديدة، يمكن أن نجد قيم مداخل القلابات تبعاً لقيم مخارجها في اللحظة الراهنة. لإيجاد ذلك نرتب في جدول قيم مخارج القلابات في اللحظة الراهنة Q والقيم التالية q، ونحدد لكل قلاب قيم المداخل J و K التي تؤدي بالقلابات إلى القيم التالية q انطلاقاً من قيمها في هذه اللحظة Q. وبتطبيق ذلك على مسألتنا نجد أن مداخل القلابات الثلاثة تأخذ القيم المبينة في الجدول التالي:

	Q_A	Q_B	Q_C	q_a	q_b	q_c		J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	0	1		0	X	0	X	1	X
1	0	0	1	0	1	0		0	X	1	X	X	1
2	0	1	0	0	1	1		0	X	X	0	1	X
3	0	1	1	1	0	0		1	X	X	1	X	1
4	1	0	0	1	0	1		X	0	0	X	1	X
5	1	0	1	1	1	0		X	0	1	X	X	1
6	1	1	0	1	1	1		X	0	X	0	1	X
7	1	1	1	0	0	0		X	1	X	1	X	1

حيث تشير Q_C و Q_B و Q_A إلى مخارج القلابات الثلاثة.

لنأخذ كمثال على أحد أسطر هذا الجدول السطر الثاني. نلاحظ أن القلاب الأول ينتقل من القيمة 0 إلى القيمة 0. مداخل القلاب في هذه الحالة هي: $J_A = 0$ أما K_A فيمكن أن يأخذ القيم 1 أو 0. أما القلاب الثاني فينتقل من الصفر إلى الواحد، ومن ثم فالمدخل J_B يجب أن يكون مساوياً للواحد، أما المدخل K_B فيمكنه أن يأخذ القيم 1 أو 0، وهكذا...

نشتق من الجدول السابق الجداول التالية الخاصة بمتغيرات دخل

كل من القلابات الثلاثة:

- القلاب J_A , K_A

Q_C \ Q_AQ_B	00	01	11	10
0	0	0	X	X
1	0	1	X	X

J_A

Q_C \ Q_AQ_B	00	01	11	10
0	X	X	0	0
1	X	X	1	0

K_A

ومنه المعادلات:

$$J_A = K_A = Q_B.Q_C$$

- القلاب J_B , K_B

Q_C \ Q_AQ_B	00	01	11	10
0	0	X	X	0
1	1	X	X	1

J_B

Q_C \ Q_AQ_B	00	01	11	10
0	X	0	0	X
1	X	1	1	1

K_B

ومنه المعادلات:

$$.J_B = K_B = Q_C$$

- القلاب J_C , K_C

Q_C \ Q_AQ_B	00	01	11	10
0	1	1	1	1
1	X	X	X	X

J_C

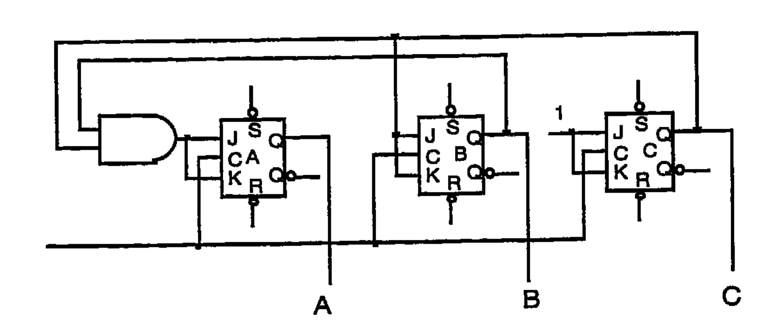
Q_C \ Q_AQ_B	00	01	11	10
0	X	X	X	X
1	1	1	1	1

K_C

ومنه المعادلات:

$$J_C = K_C = 1$$

من جملة المعادلات السابقة نجد مخطط التوصيل التالي:



(المداخل S و R هي لوضع مخارج القلاب على القيمة 1 أو 0 في لحظة البدء.)

<u>تمرين</u>: المطلوب بناء عداد تظهر على مخارجه الأرقام من 0 إلى 9 بالترميز الطبيعي، وذلك باستخدام قلابات JK.

<u>مثال</u>: تصميم العدادات باستخدام القلابات D

بالطريقة نفسها المتبعة آنفاً، يمكن تصميم عدادات باستخدام القلابات D.

ليكن المطلوب تنفيذ عداد باستخدام قلابات من نوع D، يظهر على مخرجيه Q_B و Q_A التتالي الآتي:

Q_A	Q_B
0	1
1	1
1	0

الجدول المرافق للوضع القادم للعداد بدلالة الوضع الحالي هو:

Q_A	Q_B	q_a	q_b
0	1	1	1
1	1	1	0
1	0	0	1

ومنه نجد قيم مداخل القلابات كما يلي:

Q_A	Q_B	q_a	q_b	Da	Db
0	1	1	1	1	1
1	1	1	0	1	0
1	0	0	1	0	1

ومنه جداول كارنو التي تربط مخارج العداد في اللحظة الراهنة بمخارجه في اللحظة التالية:

Q_B \ Q_A	0	1
0	X	0
1	1	1

Da

Q_B \ Q_A	0	1
0	X	1
1	1	0

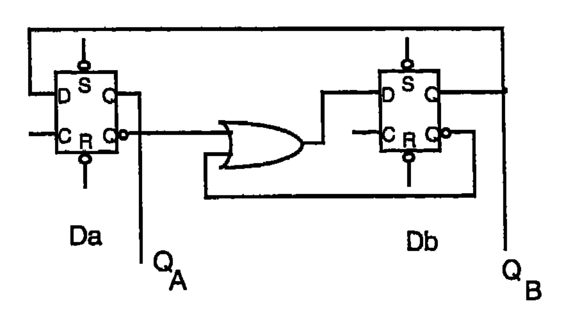
Db

ومنه معادلات مداخل القلابات:

$Da = q_b$

$Db = \overline{q_a} + \overline{q_a}$

يمكن بناءً على المعادلات المذكورة آنفاً إنشاء المخطط التقاني للعداد:



3-5 سجلات الانزياح

تستخدم سجلات الانزياح Shift Registers كذاكرة لكلمة تخزن فيه إما تسلسلياً (خانة بعد خانة) أو تفرعياً (تخزين الكلمة دفعة واحدة)، وتبقى الكلمة مخزنة حتى قراءتها التي يمكن أن تجري إما تسلسلياً

أو تفرعياً بحسب الحاجة. يمكن استبقاء المعلومات المخزنة أو التخلص منها بعد قراءتها أو تعديلها بكلمة جديدة. تستخدم القلابات المتزامنة بمختلف أنواعها في بناء السجلات، وبوجه خاص القلاب D.

نعرف طول الكلمة المراد تخزينها في السجل بعدد الخانات الاثنانية المراد تسجيلها.

تصنف سجلات الانزياح تبعاً لشكل إدخال المعلومات وقراءتها على النحو التالي:

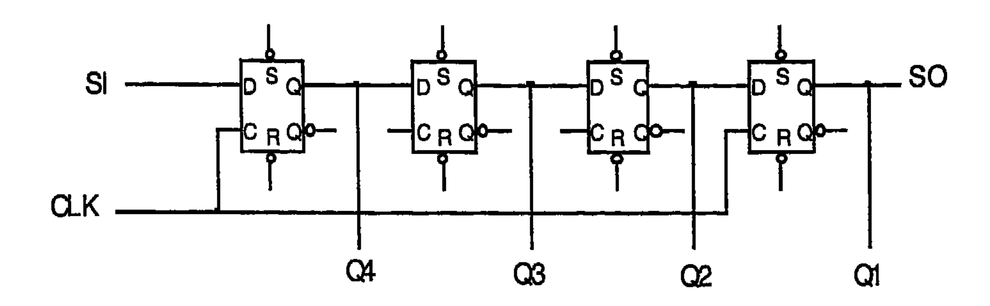
• دخل تسلسلي وخرج تسلسلي: أي إن خانات الكلمة تدخل إلى السجل خانة بعد خانة وتظهر على خرجه بالترتيب الذي أُدخلت فيه، وذلك بعد عدد من نبضات الساعة يساوي إلى طول الكلمة.

• دخل تسلسلي وخرج تفرعي: أي إن الخانات تدخل إلى السجل خانة بعد خانة وتظهر على خرجه دفعة واحدة، بعد عدد نبضات ساعة مساوٍ لطول الكلمة.

• دخل تفرعي وخرج تسلسلي: أي إن الكلمة تُسجَّل كاملة في لحظة واحدة ثم تتابع قراءتها مع النبضة التالية، فتظهر على خرجه الخانة الأولى في لحظة التسجيل.

• دخل تفرعي وخرج تفرعي: أي إن الكلمة تُسجَّل كاملة في لحظة ما وتُقرأ على الخرج في اللحظة نفسها. وتبقى الكلمة على دخله السجل حتى اللحظة التي نودّ فيها تغييرها.

وفيما يلي مثال عن سجل انزياح مبني باستخدام قلابات D بسعة أربع خانات. الدخل SI تسلسلي والخرج بالشكلين التفرعي والتسلسلي. الخرج التفرعي نجده في النقاط المشار إليها بـ Q1, Q2, Q3, Q4 حيث نجد الكلمة كاملة بعد أربع نبضات ساعة CLK. نجد في اليمين الخانة التي أدخلت أولاً إلى السجل. وتظهر على الخرج التسلسلي SO الخانة التي أُدخلت أولاً بعد أربع نبضات ساعة. وفي النبضة الخامسة للساعة نجد الخانة التي أدخلت إلى السجل في النبضة الثانية، وهلم جرا...



سنستعرض في الفصل القادم بشيء من التفصيل الدارات التتابعية المتزامنة. أما الدارات اللامتزامنة فتخرج عن إطار هذا الكتاب.

الفصل الخامس

الدارات التتابعية المتزامنة

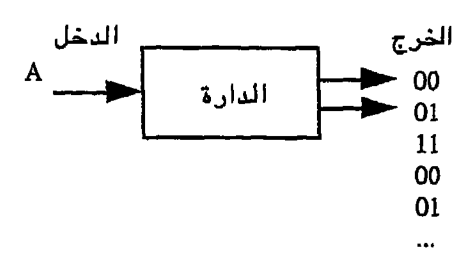
1 مقدمة

ذكرنا في الفصل السابق أن الدارات التتابعية المتزامنة هي دارات تتابعية خرجها في لحظة ما يتعلق بدخلها الخارجي، ويتعلق كذلك بالمتغيرات الداخلية أو متغيرات الحالة في تلك اللحظة. يضاف إلى ذلك أن التغيرات في الخرج والمتغيرات الداخلية محكومة بإشارة متغيرة بين الصفر والواحد ودورية في الغالب، نسميها الساعة. ولا يمكن للخرج والمتغيرات الداخلية أن تغير من قيمتها إلا مرة، ومرة واحدة خلال دور الساعة. ولهذا تحافظ المتغيرات الداخلية والخرج على قيمتها أثناء دور الساعة ولو تغير الدخل الخارجي أكثر من مرة، ولا تتغير تلك القيم إلا أثناء انتقال إشارة الساعة من الصفر إلى الواحد (الجبهة الصاعدة)، أو من الواحد إلى الصفر (الجبهة الهابطة)، أو عندما تكون إشارة الساعة على القيمة واحد، أو القيمة صفر. فالجملة التتابعية المتزامنة هي إذن دائماً في وضع مستقر، وتحدث التغيرات عند الانتقال من وضع مستقر إلى وضع مستقر آخر، وفق إيقاع الساعة.

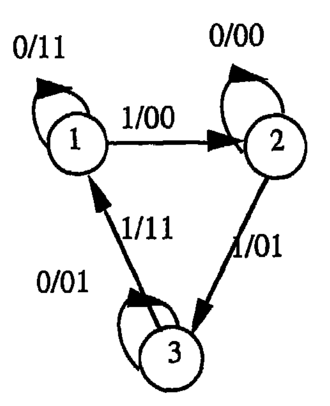
لنتأمل المثال التالي:

على جملة منطقية أن تعطي على خرجها التسلسل التالي: أولاً 00

ثم 01 ثم 11 وبعدها 00 مادامت إشارة الدخل A تساوي الواحد، وفي حال أخذ الدخل A القيمة صفر تحافظ الدارة على خرجها السابق حتى يصبح الدخل A مساوياً للواحد من جديد لتتابع الدارة توليد التسلسل المطلوب.



من الواضح أن تحقيق التتالي المطلوب لن يحدث إلا إذا افترضنا وجود إشارة دخل إضافية هي الساعة تجري التغيرات في الخرج على إيقاعها. ومنه نجد مخطط الأوضاع والانتقالات التالي:



حيث تشير الدوائر إلى الأوضاع المختلفة للدارة، التي توجد فيها بعد انتهاء أثر الساعة، أما الخطوط الموجهة فتشير إلى الانتقالات بين الأوضاع.

يظهر على هذه الخطوط قيمة الدخل وإلى يمينها قيمة الخرج الموافقة. فإذا كنا في الوضع 1 مثلاً يكون الخرج 11، وعند قدوم نبضة ساعة نبقى في الوضع 1 إذا كان الدخل مساوياً للصفر أو ننتقل إلى الوضع 2 إذا كان الدخل مساوياً للواحد، ويصبح الخرج عندها مساوياً

00، ونبقى في هذا الوضع في النبضة القادمة إذا كان الدخل معدوماً أو نتابع المسير إلى الوضع 3، وهكذا...

لنلاحظ أن الساعة كدخل لم تظهر صراحة في مخطط الانتقالات، وإنما ضمناً. فعند الانتقال من وضع إلى آخر أو البقاء في الوضع نفسه يظهر التغير في الخرج بدلالة الدخل، وهذا الانتقال يجري تبعاً لإيقاع الساعة الخارجية.

2 تصميم الدارات التتابعية المتزامنة

يتّبع في تصميم الدارات التتابعية المتزامنة الخطوات التالية، مع ملاحظة أننا نستخدم في المقام الأول القلابات المتزامنة (ذلك أننا لم نظهر وضوحاً الساعة كدخل خارجي):

1 تحديد مخطط الأوضاع والانتقالات انطلاقاً من نص المسألة بعد مناقشته من مختلف جوانبه.

2 تحديد جدول الانتقالات من مخطط الأوضاع والانتقالات، وكذلك جدول الخرج المرافق.

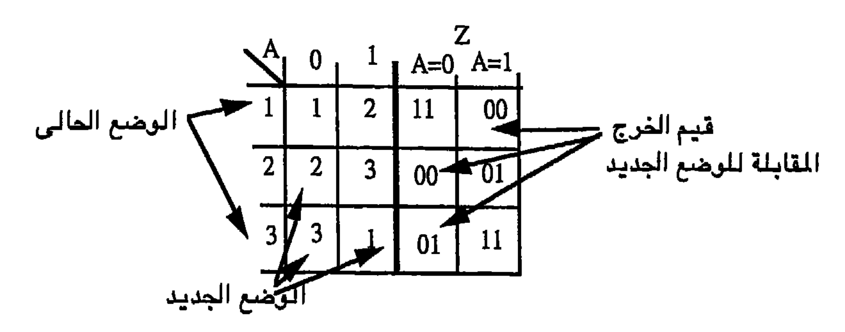
3 تحديد الأوضاع المتكافئة، وهي الأوضاع التي تقود إلى أوضاع جديدة متماثلة أومتكافئة ولها الخرج نفسه، واختزال جدول الانتقالات إن أمكن.

4 ترميز جدول الانتقالات.

5 تحديد معادلات الحالة تبعاً لنوع القلابات التي ستسخدم في تنفيذ الدارة.

6 تنفيذ الدارة والتوثق من أدائها.

جدول الانتقالات والخرج بالنسبة لمثالنا السابق هو:



حيث Z هو الخرج.

لا توجد في هذا الجدول أوضاع متكافئة. أي إن الانتقال من الوضع 1 مثلاً لن يؤدي إلى الأوضاع التي سننتقل إليها من الوضع 2 أو الوضع 3.

نعطي للوضع 1 الترميز 11 وللوضع 2 الترميز 00، وللوضع 3 الترميز 01. وقد أخذنا بهذا الترميز لتطابقه مع الخرج، وهو أمر ليس ضرورياً ولكنه يختصر من معادلات الخرج، ومن ثم من عدد البوابات التي ستسخدم في تنفيذ الدارة.

نسمي X و Y متغيري الحالة اللذين يختزنان ما طرأ على الجملة في الماضي (ذاكرة)، لنحصل بذلك على الجدول التالي، الذي نسميه جدول الحالة:

XY \ A	0	1
00	00	01
01	01	11
11	11	00
10	XX	XX

حيث نشير بالحرف X إلى أوضاع عدم التعيين.

لنلاحظ أن هذا الترميز قد أدى إلى تغيير في ترتيب الأسطر وذلك لتحقيق التجاور، فالسطر الأول في جدول الانتقالات أصبح السطر الثالث في جدول الحالة، في حين أصبح السطر الثاني في

جدول الانتقالات السطر الأول في جدول الحالة، والسطر الثالث في جدول الانتقالات أصبح الثاني في جدول الحالة. وبذلك يكون ترميز الحالة مطابقاً لترميز الخرج.

ملاحظة هامة: لنتذكّر أن الانتقال من قيم متحولات الحالة المبينة إلى يسار الجدول والقيم داخل الجدول وعلى السطر نفسه لا يحدث إلا بعد النبضة التالية للساعة، وهذا ما ينطبق على الخرج أيضاً. أما قيم متحولات الحالة داخل الجدول فتصبح مساوية للقيم الموجودة في يساره بعد برهة δt أقصر من دور الساعة.

نجد معادلات الحالة:

$$x = A.Y.\overline{X} + \overline{A}.X$$

$$y = \overline{A}.Y + A.\overline{Y} + \overline{X}.Y$$

حيث يشير الحرف الكبير X إلى قيمة متغير الحالة في اللحظة قبل ورود نبضة الساعة؛ أما الحرف الصغير x فيشير إلى قيمة متغير الحالة في اللحظة التالية لقدوم نبضة الساعة التالية. ويقال مثل ذلك عن متغير الحالة الثاني. لنلاحظ أن الخرج ينطبق على الحالة XY وذلك بسبب الخيار الذي اعتمدناه لترميز جدول الانتقالات.

3 تنفيذ الدارات التتابعية باستخدام القلابات

1-3 التنفيذ باستخدام القلابات JK

لنتذكر أن معادلة القلاب JK هي:

$$q = J.\overline{Q} + \overline{K}.Q$$

حيث J, K مداخل القلاب و q هو خرجه بعد أن تصبح الساعة فعالة، أما Q فهو خرج القلاب قبل أن تصبح الساعة فعالة.

وكما تُظهر المسألة السابقة، نحتاج إلى قلابين ليرتبط كل متحول من متحولات الحالة بإحد القلابين، وما علينا تحديده الآن هو مداخل القلابين. لذلك نعيد صياغة المعادلتين أعلاه وفقاً لشكل معادلة

القلاب JK على النحو التالي:

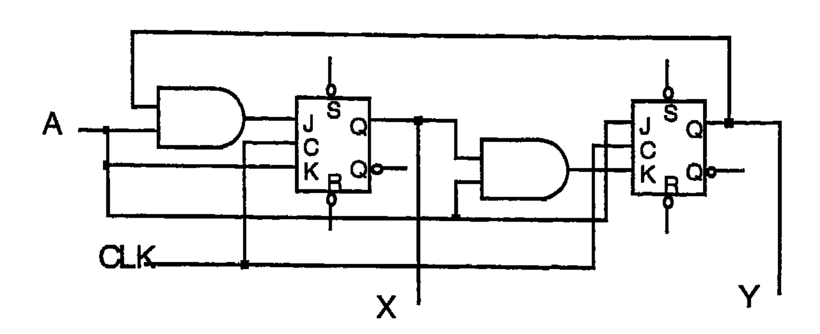
$$x = A.Y.\overline{X} + \overline{A}.X$$

$$y = A.\overline{Y} + (\overline{A} + \overline{X}).Y$$

وبالمطابقة مع معادلة القلاب نجد:

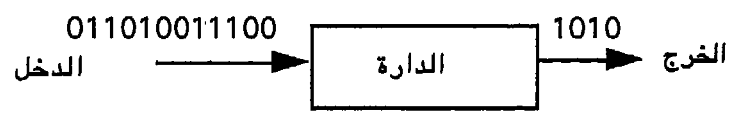
$$Jx = A.Y,\ Kx = A,\ Jy = A,\ Ky = A.X$$

ومنه مخطط الدارة:



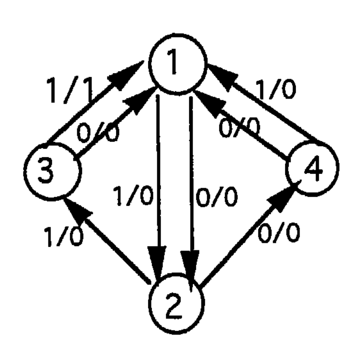
مثال:

تستقبل دارة منطقية رسائل مؤلفة من ثلاث خانات تأتي تباعاً على خط إرسال وحيد، على هذه الدارة أن تعطي على خرجها Z القيمة 1 كلما تحقق التالي $E(t\text{-}2) = X, E(t\text{-}1) = 1, E(t) = 1$ (حيث t هي اللحظة الراهنة، و t-1 اللحظة السابقة، و t-2 اللحظة الأسبق)، أي كلما كان آخر خانتين من رسالة ثلاثية الخانات مساويتين للواحد، وذلك مهما تكن قيمة الخانة الأولى. يشير الحرف X إلى أن الدخل قد يكون واحداً أو صفراً.



هذه الدارة تتابعية كما يشير إلى ذلك نص المسألة، وهي متزامنة لعدم إمكان التمييز بين خانتين متتاليتين حين تساويهما بالقيمة إلا عن طريق ساعة خارجية.

يعطى مخطط الأوضاع والانتقالات كما يلي:



حيث ننتقل من الوضع 1 إلى الوضع 2 مع أول نبضة ساعة مهما تكن قيمة الدخل، بعد ذلك نذهب من الوضع 2:

- إما إلى الوضع 4 إذا كان الدخل مساوياً للصفر، لنعود بعد ذلك إلى الوضع 1 وبخرج معدوم أياً كانت قيمة الدخل عند الانتقال من 4 إلى 1؛

- أو إلى الوضع 3 إذا كان الدخل مساوياً 1، ونعود بعد ذلك إلى الوضع 1 بخرج معدوم إذا كان الدخل معدوماً أو بخرج يساوي 1 إذا كان الدخل مساوياً للواحد.

الانتقال بين وضع وآخر يجري تبعاً لإيقاع الساعة.

من المخطط السابق نجد جدول الانتقالات والخرج التالي:

A	0	1	Z 0	Z 1
1	2	2	0	0
2	4	3	0	0
3	1	1	0	1
4	1	1	0	0

ولا توجد أوضاع متكافئة وضوحاً.

لنعط للوضع 1 الترميز 00، وللوضع 2 الترميز 01، وللوضع 3 الترميز 11، وأخيراً للوضع 4 الترميز 10. وليكن X و Y متغيري الحالة.

نجد جدول الحالة:

XY \ A	0	1
00	01	01
01	10	11
11	00	00
10	00	00

وعليه فمعادلات الحالة هي:

$x = Y.\overline{X}$
$y = \overline{X}.\overline{Y} + A.\overline{X}$

أو بالشكل الذي يقابل صيغة القلاب JK:

$x = Y.\overline{X}$
$y = \overline{X}.\overline{Y} + A.\overline{X}.Y$

ومنه:

$Jx = Y, Kx = 1, Jy = \overline{X}, Ky = \overline{A} + X$

أما جدول الخرج فهو:

XY \ A	0	1
00	0	0
01	0	0
11	0	1
10	0	0

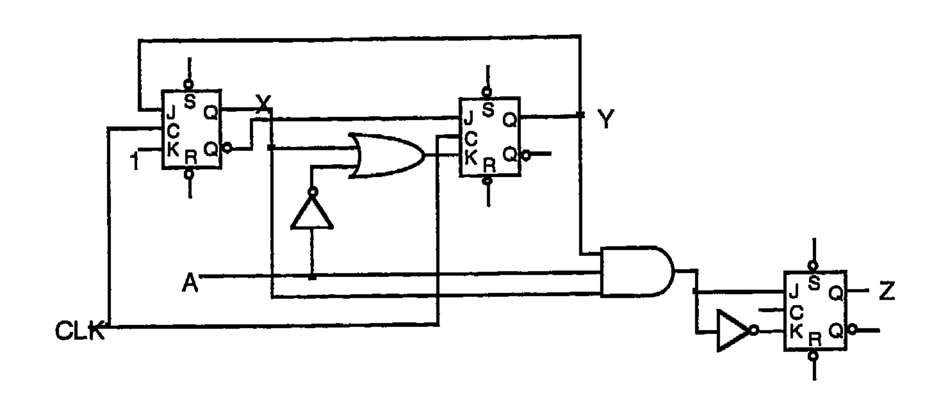
ومنه فمعادلة الخرج:

$Z = A.X.Y$

لنلاحظ أن الخرج لن يأخذ القيمة 1 بمجرد أن تصبح مركبتا الحالة مساويتين للواحد، وإنما بعد نبضة جديدة للساعة كما يبين ذلك جدولا الحالة والخرج، على هذا يجب إلحاق الخرج بقلاب إضافي توصل مداخله كما يلي:

$Jz = A.X.Y, Kz = \overline{A.X.Y}$

ومنه مخطط الدارة المطلوبة:



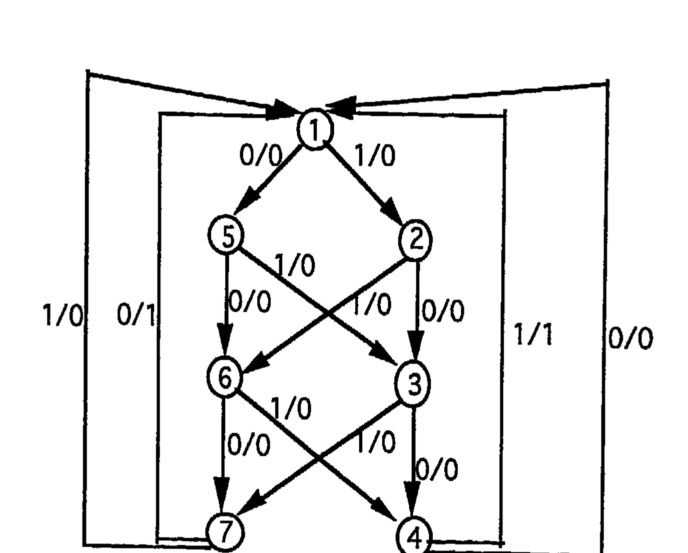
2-3 التنفيذ باستخدام القلابات D

نستعرض الآن طريقة التنفيذ باستخدام القلابات من نوع D، وذلك بتأمل المثال التالي:

على دارة منطقية أن تقوم باختبار لكل أربع خانات ثنائية متتالية، رباع رباع، بحيث تعطي على خرجها القيمة 1 كلما كان عدد القيم المساوية للواحد في الخانات الأربع التي يطبق عليها الاختبار زوجياً، أو كانت كل الخانات مساوية للصفر.

هل الدارة اللازمة لتنفيذ ذلك من النوع المتزامن أم لا؟الجواب بنعم ذلك أنه لن يمكننا تعرف خانتين متساوتين ومتتاليتين بدون إشارة إضافية للتمييز، وهي الساعة.

مخطط الأوضاع والانتقالات:



ومنه جدول الانتقالات والخرج:

A	0	1	Z 0	Z 1
1	5	2	0	0
2	3	6	0	0
3	4	7	0	0
4	1	1	0	1
5	6	3	0	0
6	7	4	0	0
7	1	1	1	0

لذا نحتاج إلى ثلاثة قلابات، ذلك أننا نحتاج إلى ثلاثة متغيرات حالة.

من ترميز جدول الانتقالات يمكن استنتاج مداخل القلابات D كما في الجدول التالي:

XYZ \ A	0	1	Da 0	Da 1	Db 0	Db 1	Dc 0	Dc 1	Z :output 0	Z :output 1
000	110	001	1	0	1	0	0	1	0	0
001	011	111	0	1	1	1	1	1	0	0
011	010	101	0	1	1	0	0	1	0	0
010	000	000	0	0	0	0	0	0	0	1
110	111	011	1	0	1	1	1	1	0	0
111	101	010	1	0	0	1	1	0	0	0
101	000	000	0	0	0	0	0	0	1	0
100	XXX	XXX	X	X	X	X	X	X	X	X

ومنه معادلات القلابات:

$$Da = (\overline{Y}.\overline{Z}+X.Y).\overline{A} + \overline{X}.Z.A$$
$$Db = (\overline{X}.\overline{Y} + \overline{X}.Z).\overline{A} + \overline{X}.\overline{Y}.Z + X.\overline{Z} + X.Y.A$$
$$Db = (\overline{X}.\overline{Y} + \overline{X}.Z).A + \overline{X}.\overline{Y}.Z + X.\overline{Z} + X.Y.\overline{A}$$

أما معادلة الخرج فهي:

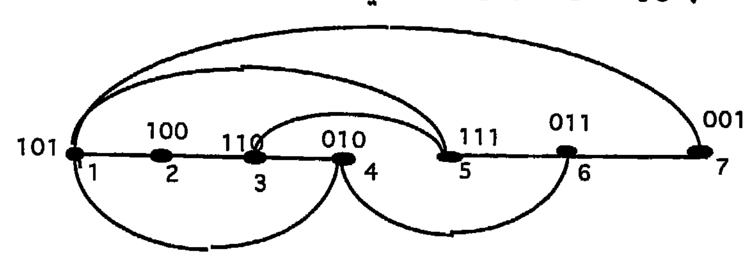
$$W = \overline{X}.Y.\overline{Z}.A + X.\overline{Y}.\overline{A}$$

وعلى أساس المعادلات السابقة، يمكن توصيل القلابات الثلاثة مع ملاحظة أننا نحتاج أيضاً إلى قلابٍ للخرج.

نلاحظ في المثال السابق كثرة عدد البوابات اللازمة لتحقيق الدوال الخاصة بمداخل القلابات، حتى لو أخذنا بعين الاعتبار وجود حدود مشتركة بين الدوال المنطقية الخاصة بهذه المداخل. من الطبيعي إذن أن نوجه لأنفسنا السؤال التالي: هل يمكن تنفيذ المسألة بعدد أقل من البوابات والقلابات؟ لنلاحظ أولاً الفصل بين عدد البوابات المنطقية وعدد القلابات، فعدد القلابات مرتبط بعدد متحولات الحالة أو بعدد الأوضاع الممثلة للجملة، ومن ثم يجب البحث عن أقل عدد أوضاع كاف لتحقيق المسألة، أما عدد البوابات المنطقية فهو مرتبط بترميز جدول الأوضاع، لهذا يجب استخدام أفضل ترميز ممكن لجدول الأوضاع.

4 ترميز جدول الأوضاع

اتبعنا فيما سبق طريقة مباشرة في ترميز الأوضاع، إلا أن المسألة ليست دائماً بهذه السهولة. بوجه عام، نبحث عن ترميز يأخذ بعين الاعتبار التجاورات بين الأوضاع، بناء على مخطط يربط الأوضاع والتجاورات فيما بينها. وفي حالة مثالنا السابق نجد مخطط التجاورات والترميز التالي:



الذي يعطي جدول الترميز التالي:

XYZ \ A	0	1	Da 0	Da 1	Db 0	Db 1	Dc 0	Dc 1	Z :output 0	Z :output 1
000	XXX	XXX	X	X	X	X	X	X	X	X
001	101	101	1	1	0	0	1	1	1	0
011	001	010	0	0	0	1	1	0	0	0
010	101	101	1	1	0	0	1	1	0	1
110	010	001	0	0	1	0	0	1	0	0
111	011	110	0	1	1	1	1	0	0	0
101	111	100	1	1	1	0	1	0	0	0
100	110	011	1	0	1	1	0	1	0	0

ومن ثم تكون لدينا المعادلات التالية:

$$Da = \overline{X}\,\overline{Y} + \overline{X}\,\overline{Z} + AXZ + \overline{A}X\overline{Y}$$

$$Db = \overline{A}.X + AYZ + \overline{Y}\,\overline{Z}$$

$$Dc = \overline{A}\,\overline{X} + \overline{A}Z + AYZ + \overline{X}\,\overline{Y}$$

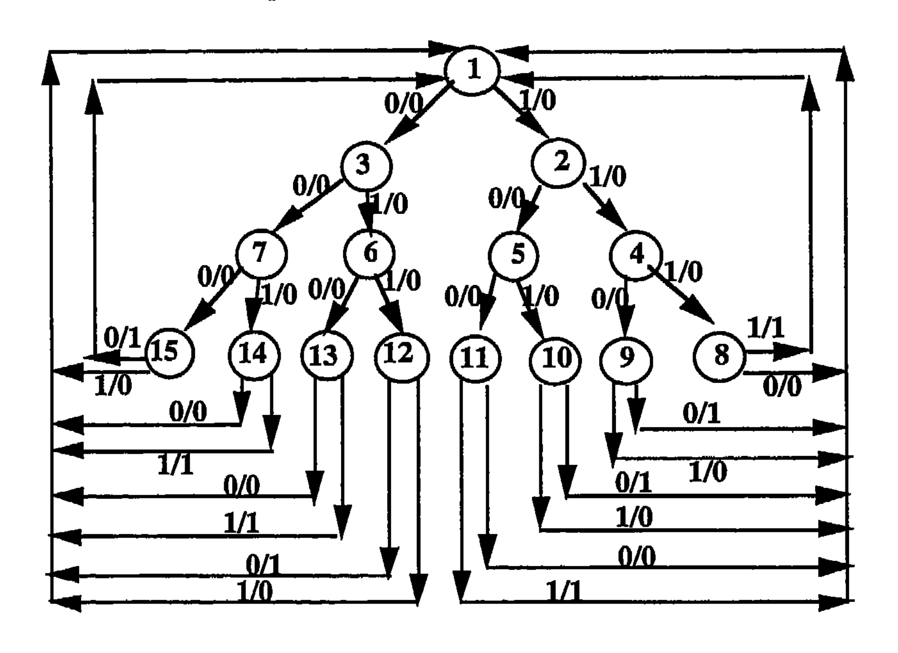
أما معادلة الخرج فهي:

$$W = \overline{A}.\overline{X}.\overline{Y} + A.\overline{X}.\overline{Z}$$

5 أوضاع التكافؤ

علينا أن نسأل أنفسنا أمام كل مسألة عن العدد الأصغر من الأوضاع اللازم لحل المسألة المطروحة، لأن ذلك يقودنا إلى أقل عدد من متغيرات الحالة، أي إلى أقل عدد لازم من القلابات. والسؤال الآن: كيف نصل إلى هذا العدد الأصغر أياً كان تمثيل المسألة بمخطط الأوضاع والانتقالات (والذي نفترضه بدون أخطاء طبعاً)؟ الجواب في حذف الأوضاع المتكافئة الذي لن ندخل في تفاصيله، ولكن سنوضحه بالعودة إلى المثال السابق.

لنلاحظ أنه كان من الممكن وضع مخطط مختلف للانتقالات بحيث يكون لدينا خمسة عشر وضعاً على النحو التالي:



حيث عولجت هنا كل امكانية على حدة.

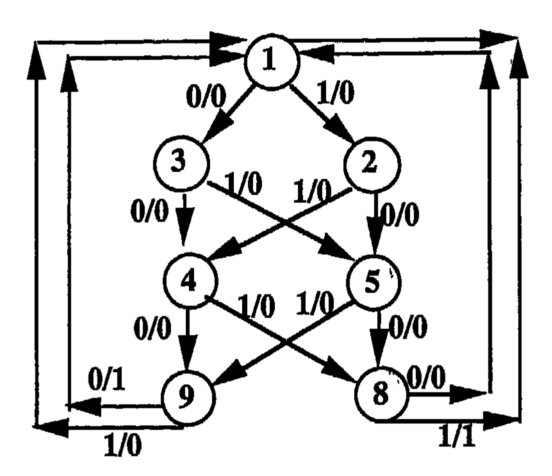
من مخطط الأوضاع والانتقالات هذا نجد جدول الانتقالات التالي:

	الوضع التالي		الخرج	
A	0	1	0	1
1	3	2	0	0
2	5	4	0	0
3	7	6	0	0
4	9	8	0	0
5	11	10	0	0
6	13	12	0	0
7	15	14	0	0
8	1	1	0	1
9	1	1	1	0
10	1	1	1	0
11	1	1	0	1
12	1	1	1	0
13	1	1	0	1
14	1	1	0	1
15	1	1	1	0

نلاحظ من الجدول السابق أن الأوضاع 8 و 10 و 12 و 15 هي أوضاع متكافئة، وتكوّن صف تكافؤ، بمعنى أن لها الأوضاع القادمة نفسها وكذلك الخرج في حالة الدخل نفسه. وبالمشابهة، فالأوضاع 9 و 11 و 13 و 14 هي أوضاع متكافئة وتكوّن فيما بينها صف تكافؤ. وينجم عن هذا تكافؤ الأوضاع 4 و 6 التي تكوّن صف تكافؤ أيضاً، وكذلك تكافؤ الأوضاع 5 و 7 التي تكوّن صف تكافؤ آخر. ومن ثم نحصل على جدول أوضاع مكافئ هو:

	الوضع التالي		الخرج	
A	0	1	0	1
1	3	2	0	0
2	5	4	0	0
3	4	5	0	0
4	9	8	0	0
5	8	9	0	0
8	1	1	0	1
9	1	1	1	0

وهذا يقود إلى مخطط الأوضاع والانتقالات التالي المكافئ للمخطط الأول:





الجزء الثاني

المعالجات الصغرية

وبنيان الحواسيب



الفصل الأول

وحدة المعالجة المركزية – المعالج 8086

1 مقدمة

تدل لفظة حاسوب، تبعاً لاشتقاقها اللغوي، على آلة مهمتها الحساب، ولكن استخدامات الحاسوب العملية تتجاوز هذا المعنى الأولي بكثير. ومع أنه ليس من اليسير إعطاء تعريف دقيق لماهية الحاسوب، فإنه يمكن مبدئياً الاصطلاح على أنه «آلة إلكترونية قادرة على تداول المعلومات (ذات الطبيعة الرقمية) بسرعة كبيرة، بهدف معالجتها أو تنسيقها أو تخزينها أو نقلها، وذلك عن طريق تنفيذ برنامج يحدد بدقة خطوات العمل المطلوب إنجازه».

تختلف طبيعة ا لمعلومات التي يتداولها الحاسوب باختلاف المجال أو الحقل الذي يُستخدم فيه. فقد تكون تلك المعلومات عددية، مثل المعلومات العلمية والهندسية، أو الإحصائية والمالية؛ أو قد تكون على هيئة نصوص نريد معالجتها ونشرها، أو تخزينها واسترجاعها عند الطلب؛ أو على هيئة بيانيات أو صور أو أصوات نريد معالجتها أو تعرّفها أو عرضها؛ أو على شكل إشارات إلكترونية، كالقياسات

وإشارات التحكم في المنظومات الصناعية وما أشبه. وفي جميع الأحوال، يجب أن تُقدَّم المعلومات للحاسوب بصيغة رقمية، وتجري معالجة تلك المعلومات، كما سنرى، باستخدام مجموعة من العمليات الحسابية والمنطقية يختلف حجمها ودرجة تعقيدها باختلاف الحاسوب المستخدم.

يظهر مما سبق أن الحاسوب آلة متعددة التطبيقات يمكن استخدامها في مجالات متنوعة من أجل أداء مهمات مختلفة. والسؤال الذي يطرح نفسه هنا هو: كيف يمكن «تخصيص» عمل الحاسوب تبعاً للمهمة الموكلة إليه؟ بمعنى آخر: كيف يمكن أن نُعرِّف للحاسوب الوظيفة التي نريد منه تنفيذها في لحظة معينة؟ لا بد إذن من وجود وسيلة لتزويد الحاسوب بالطريقة الواجب اتباعها لإنجاز المهمة. تسمى تلك الوسيلة برمجة الحاسوب، وتتم عن طريق تزويد الحاسوب ببرنامج يتكوّن من سلسلة من التعليمات التي يستطيع الحاسوب فهمها، والتي تحدد بدقة تتابع الخطوات المؤدية بمجموعها إلى تنفيذ العمل المطلوب.

باختصار، نقول أن الحاسوب يتكون من منظومة من التجهيزات الإلكترونية المادية (أو الكيان الصلب) التي يمكن أن تُنفَّذ عليها مجموعة كبيرة من الوظائف المتنوّعة؛ ويجري تعريف وتحديد الوظيفة المطلوبة في لحظة معينة بواسطة عمليات البرمجة. نطلق عادة على منظومة البرامج العاملة على الحاسوب اسم البرمجيات (أو الكيان الليّن).

تعرّف الطلاب في مادة سابقة بنية الحاسوب ومكوّناته الأساسية. أما هذه المادة، فالغرض منها الدخول في تفاصيل تلك المكوّنات. وسنبدأ في هذا الفصل بدراسة وحدة المعالجة المركزية، آخذين عنها مثالاً المعالج الصغري 8086 من شركة Intel. وقد اخترنا هذا المعالج لأنه السلف الأول لعائلة المعالجات 80x86 (التي تنتمي إليها المعالجات 80286، و 80386 و 80486 و Pentium) والتي تكوّن، كما هو معروف،

لبّ الحواسيب الشخصية PC المتوافقة مع IBM، بأجيالها المختلفة.

يقدم الفصل الثاني من هذا الجزء من الكتاب عرضاً لأهم الدارات المحيطية ودارات الدخل/الخرج المستخدمة في الحواسيب. أما الفصل الثالث، أضخم فصول الكتاب، فنخصصه لدراسة لغة المجمّع للمعالج 8086. وأخيراً نقدم في الفصلين التاليين لمحة سريعة إلى بعض تقنيات المعالجة المتطورة: المعالجة التفرعية في الفصل الرابع، وبنيان الحواسيب ذات مجموعة التعليمات الموجزة في الفصل الخامس.

2 تذكرة ببنية الحاسوب

يتألف أي حاسوب من وحدة المعالجة المركزية CPU ومن الذاكرة، ومن وحدات الدخل والخرج. وترتبط هذه المكوّنات بعضها ببعض بواسطة مجموعة من الخطوط التفرعية التي تسمى المسرى Bus. وسنصف فيما يلي عمل كل من هذه الأجزاء على حدة.

• الذاكرة: يحتاج أي حاسوب إلى نوعين من الذاكرة: الذاكرة الميتة (ROM) والذاكرة الحية (RAM)؛ وقد يستخدم أيضاً أقراص تخزين مغناطيسية أو ضوئية. الهدف الأساسي من وجود الذاكرة هو تخزين سلسلة التعليمات المراد تنفيذها (أي البرنامج)؛ أما الوظيفة الثانية للذاكرة فهي تخزين المعطيات التي سيتعامل معها الحاسوب.

• وحدة الدخل/الخرج: تسمح هذه الوحدة بتبادل المعطيات بين الحاسوب والعالم الخارجي. ومن الطرفيات الأساسية نذكر لوحة المفاتيح، والشاشة، والطابعة، والموديم. تسمى الوسائل المادية التي تربط الحاسوب بالعالم الخارجي المعابر Ports. فمعبر الدخل مثلاً يسمح للمعطيات المدخلة عبر لوحة المفاتيح أو عبر مبدّل تمثيلي/رقمي بالدخول إلى الحاسوب. اما معبر الخرج فيستخدم

لإرسال المعطيات من الحاسوب إلى الطرفيات، كشاشة العرض أو الطابعة أو المبدِّل الرقمي/التمثيلي. ويتألف المعبر، من وجهة نظر فيزيائية، من مجموعة قلابات D تسمح للمعطيات بالعبور بناءً على أمر من وحدة المعالجة المركزية.

- وحدة المعالجة المركزية: وهي النواة الأساسية في أي نظام حاسوبي، فهي تحكم عمل الحاسوب، فتجلب التعليمات الاثنانية من الذاكرة، وتفككها إلى أعمال بسيطة تستطيع تنفيذها. وتحتوي وحدة المعالجة المركزية CPU على وحدة الحساب والمنطق ALU، وهي الجزء المسؤول عن تنفيذ العمليات الحسابية كالجمع والطرح، أو العمليات المنطقية كعمليات AND, OR, XOR وعمليات الإزاحة. وتستمد الوحدة أوامرها من وحدة التحكم Control Unit. وتحوي وحدة المعالجة المركزية كذلك عداداً للعناوين، يخزن عنوان التعليمة التالية الواجب جلبها، أو عنوان المعطيات المراد قراءتها أو كتابتها. إضافة إلى ذلك، تحتوي وحدة المعالجة المركزية على سجلات عامة الاستخدام تفيد في تخزين المعطيات الثانوية.

- المسري: وهو مجموعة من الخطوط المادية المستخدمة في وصل مكونات الحاسوب الأساسية. ويتألف مما يلي:

– خطوط العناوين: وهي التي تسمح لوحدة المعالجة المركزية بعنونة الذاكرة أثناء عملية القراءة أو الكتابة. يتعلق عدد المواقع التي تستطيع وحدة المعالجة المركزية التعامل معها بعدد خطوط العناوين. فإذا كان عدد الخطوط هو N فإن عدد المواقع الممكن الوصول إليها هو 2^N. مثلاً، إذا كان لوحدة معالجة مركزية عدد خطوط قدره 16 خطاً، فإن هذه الوحدة تستطيع الوصول إلى 2^{16} (أي 65536) موقعاً.

– خطوط المعطيات: هذه الخطوط ثنائية الاتجاه، أي إنها تستخدم لنقل المعطيات من وإلى وحدة المعالجة المركزية. فعلى سبيل

المثال، يمكن لهذه الخطوط نقل المعلومات من معبر ما إلى وحدة المعالجة المركزية.

– خطوط التحكم: تستخدم وحدة المعالجة المركزية مجموعة خطوط التحكم لتأهيل عمل الذاكرة أو لتأهيل معبر من معابر الحاسوب لتبادل المعطيات. فمثلاً عند قراءة كلمة من الذاكرة، تضع وحدة المعالجة المركزية عنوان موقع الذاكرة المطلوب على خطوط العناوين، وترسل إشارة القراءة على خطوط التحكم. فتتأهل الذاكرة من جراء ذلك، وترسل محتوى ذلك الموقع على خطوط المعطيات، حيث تستطيع وحدة المعالجة المركزية قراءتها.

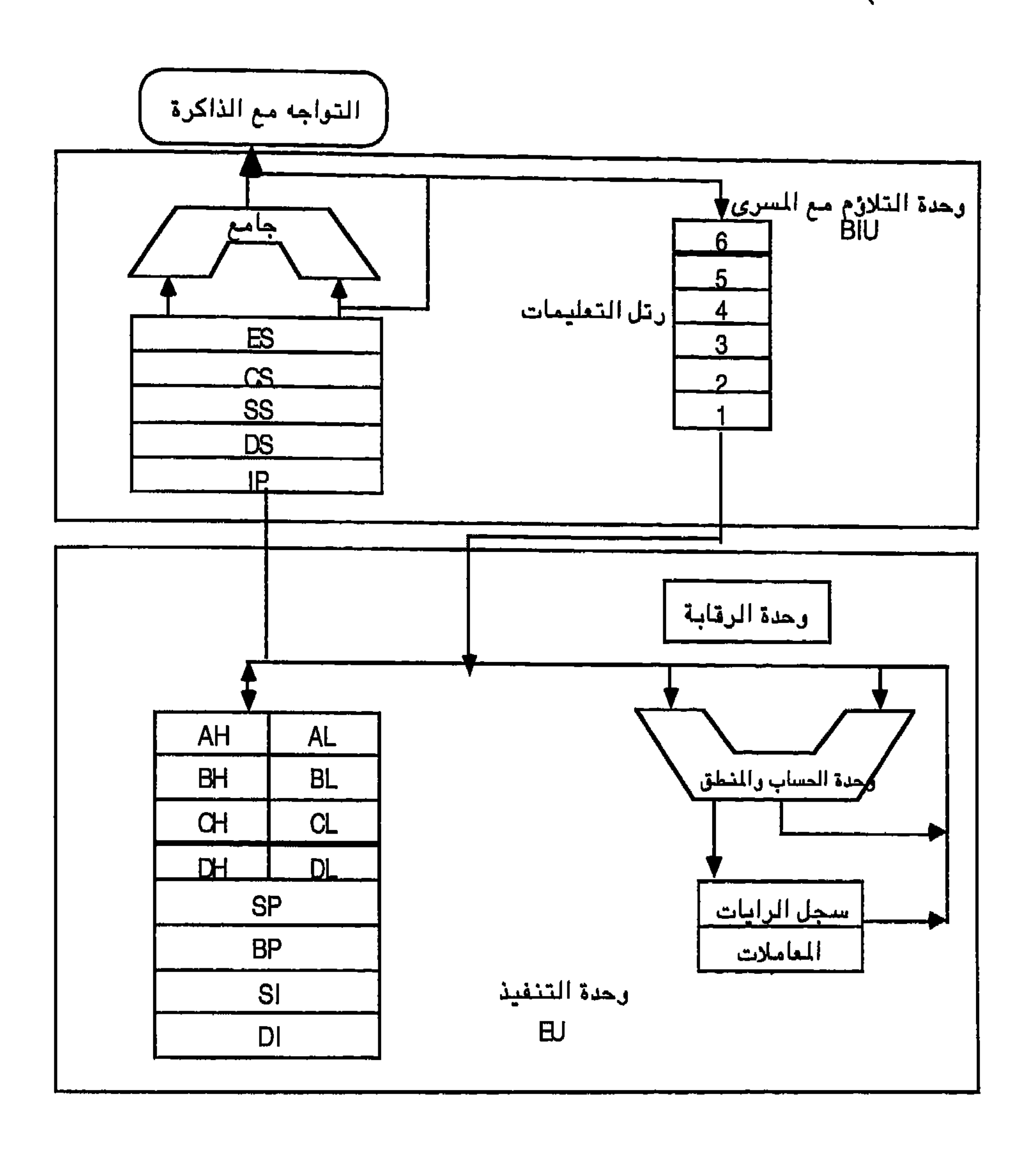
3 وحدة المعالجة المركزية للمعالج 8086

ينتمي المعالج INTEL 8086 إلى سلسلة المعالجات الصغرية ذات 16 خانة، ويقصد بذلك أن وحدة المعالجة المركزية، وسجلاته الداخلية، ومعظم تعليماته صممت لتتعامل مع الكلمات الاثنانية المرمزة على 16 خانة. فالمعالج 8086 له 16 خطاً لنقل المعطيات، فيمكنه إذن قراءة أو كتابة كلمات مرمزة على 8 خانات أو 16 خانة في الذاكرة أو في أحد المعابر.

ويحتوي مسرى المعالج على 20 خطَّ عنونة، وهذا ما يجعل عدد المواقع التي يستطع المعالج الوصول إليها هو2^{20} موقعاً (أي 576 048 1 ثمانيّة). وتخزن الكلمات المرمزة على 16 خانة في الذاكرة باستخدام موقعين متتاليين (كل موقع يحتوي على كلمة من 8 خانات). فإذا كانت الثمانيّة الأولى موجودة في موقع ذي عنوان زوجي، فإن المعالج يستطيع الوصول إلى الكلمة بتمامها بعملية واحدة. أما إذا كان عنوان الثمانيّة الأولى فردياً، فإن المعالج سيقرؤها في دور أول، ثم سيقرأ الثمانيّة الأخرى في دور آخر.

ومن الضروري، قبل كتابة برامج للمعالج الصغري 8086، فهم بنيته الداخلية ومعرفة سجلاته الداخلية. تتألف وحدة المعالجة

المركزية من جزأين مستقلين هما: وحدة التواجه مع المسرى BIU (Bus Interface Unit)؛ ووحدة التنفيذ EU (Execution Unit). ويسمح تقسيم العمل بين هاتين الوحدتين بتسريع وقع المعالجة (انظر الشكل 1).



الشكل 1: المخطط الصندوقي لوحدة المعالجة المركزية للمعالج 8086.

1-3 وحدة التواجه مع المسرى

تضع هذه الوحدة العناوين على المسرى، وتجلب التعليمات من الذاكرة، وتقرأ المعطيات من الذاكرة والمعابر، وتكتب النتائج في الذاكرة والمعابر. بمعنى آخر، تضطلع الوحدة BIU بكافة عمليات نقل المعطيات والعناوين على المسرى. وسنشرح فيما يلي عمل الأجزاء الوظيفية المكونة لهذه الوحدة.

1-1-3 الرتل

لزيادة سرعة تنفيذ البرنامج تجلب الوحدة BIU سلفاً ست تعليمات من الذاكرة وتحفظ هذه الثمانيّات الست داخل مجموعة سجلات تعمل بطريقة «الداخل أولاً يخرج أولاً» FIFO. وهذا يعني أن الثمانيّة الأولى ستقرأ أولاً لتُرسل إلى وحدة التنفيذ، تليها الثمانيّة الثانية وهكذا. تُسمى تلك السجلات الرتل Queue.

من جهة أخرى، تستطيع وحدة التواجه أن تجلب التعليمات وتخزنها في الرتل أثناء قيام وحدة التنفيذ بتفكيك التعليمة أو تنفيذها، والذي لا يتطلب استخدام المسرى.

عندما تصبح وحدة التنفيذ جاهزة للتعليمة التالية، فإن عليها أن تقرأ فقط التعليمة الجديدة من رتل وحدة التواجه، وهذا أسرع بكثير من جلبها من الذاكرة. فالجلب يتطلب وضع العناوين على المسرى وإرسال إشارة القراءة، ثم قراءة الكلمة من خطوط المعطيات. وهذه الطريقة تماثل إلى حد بعيد عمل البَنّاء. فبدلاً من أن يبحث البَنّاء كل مرة عن قطعة الأجرّ ويرصفها، يستطيع أن يضع بالقرب منه مخزوناً صغيراً من الأجرّ، يصل إليه بسرعة، ويقوم مساعده بتعبئة هذا المخزون باستمرار.

تُنفذ جميع تعليمات المعالج 8086 وفق الطريقة المذكورة، باستثناء تعليمتي القفز والاستدعاء JUMP & CALL. ففي تلك الحالة،

ينبغي إفراغ الرتل من التعليمات، ثم شحنه ثانيةً بدءاً من العنوان الجديد. وتسمى آلية جلب التعليمة الجديدة أثناء تنفيذ التعليمة الحالية بالتوارد Pipelining.

2-1-3 سجلات القطاعات

تحتوي وحدة التواجه على أربعة سجلات ذات 16 خانة تسمى سجلات القطاعات Segment Registers، وهي: سجل قطاع البرنامج CS (Code Segment)؛ وسجل قطاع المكدس SS (Stack Segment)؛ وسجل قطاع المعطيات DS (Data Segment)؛ وسجل القطاع الإضافي ES (Extra Segment). تُستخدم هذه السجلات لتخزين الجزء العلوي (16 خانة) من عناوين البداية للقطاعات الأربعة التي يحتاج إليها المعالج 8086 في تنفيذ البرنامج. فكما ذكرنا سابقاً تضع وحدة التواجه عنواناً مرمزاً على 20 خانة على خطوط العناوين، أي إنها قادرة على عنونة 576 048 1 ثمانيّة. ولما كان المعالج لا يستطيع، في لحظة معينة، أن يتعامل مع فضاء يزيد عن 65536 ثمانيّة (أي 64K ثمانيّة) فقد وجب تقسيم الفضاء الكلي إلى قطاعات. ويبين الشكل 2 القطاعات الأربعة المستخدمة.

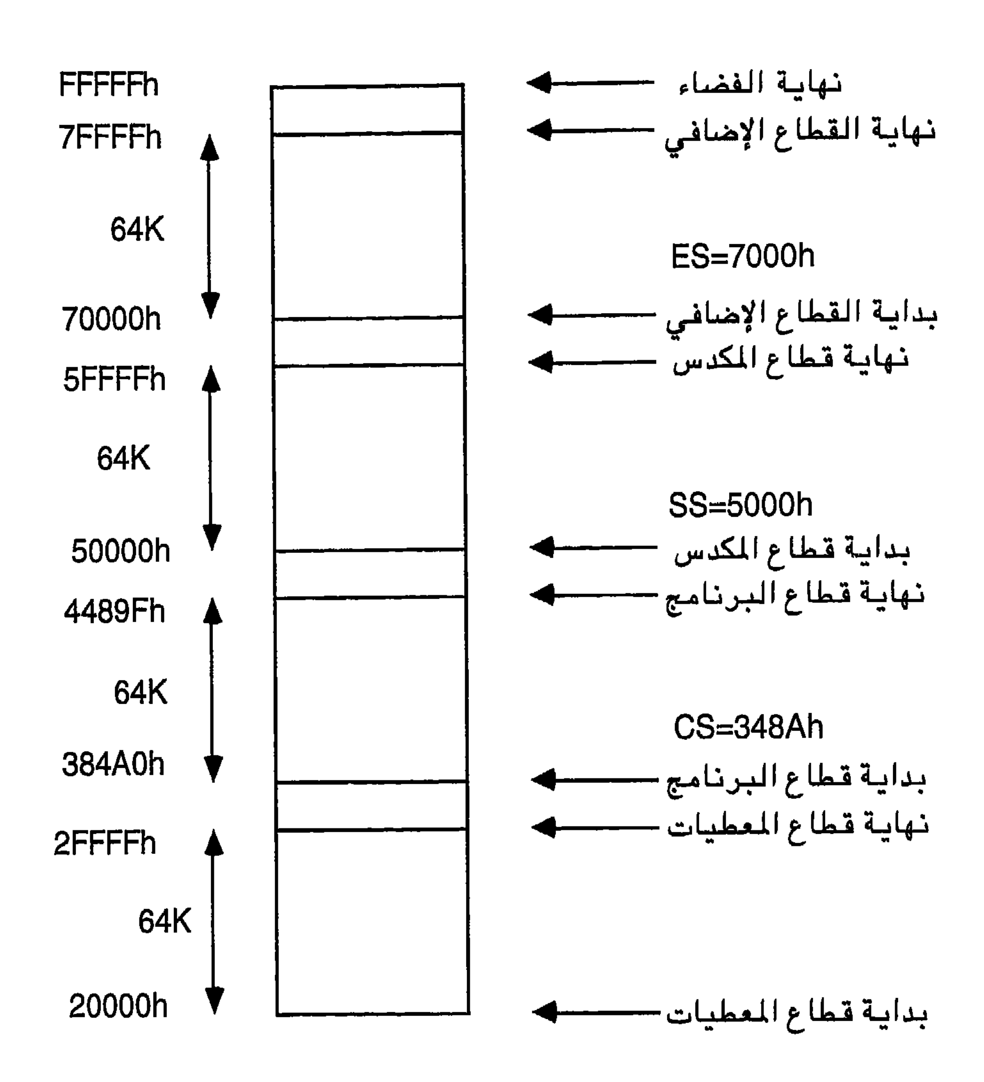
يمكن أن تكون هذه القطاعات مستقلة بعضها عن بعض، أو أن تتراكب فيما بينها. فمثلاً يمكن أن نجد في نظام ما أن القطاعات الأربعة تبدأ من العنوان ذاته 0h.

يفيد سجل قطاع البرنامج في حفظ الجزء العلوي من عنوان البداية لمكان تخزين البرنامج. ولتحديد عنوان البداية تحديداً كاملاً، تضيف وحدة التواجه أربعة أصفار في الجزء السفلي من العنوان لإتمامه إلى 20 خانة.

مثال:

إذا كان محتوى السجل CS هو 348Ah، فعنوان البداية سيكون 348A0h.

يمكن إذن للقطاعات أن تبدأ من أي موقع في الذاكرة بشرط أن ينتهي عنوان ذلك الموقع بأربعة أصفار.

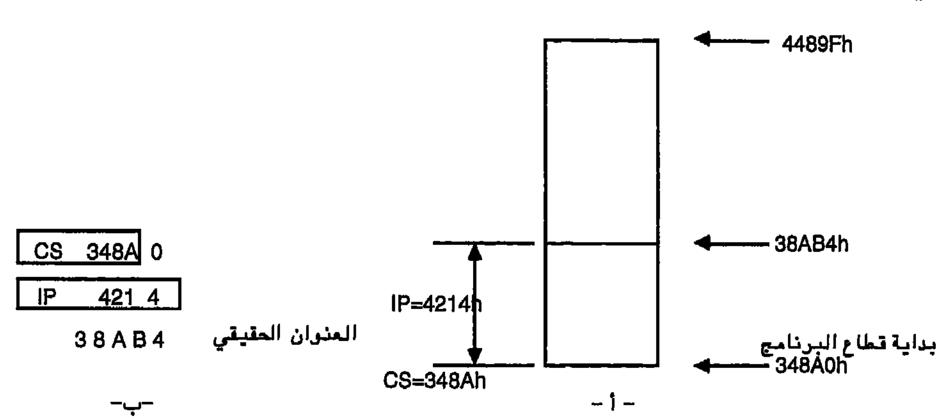


الشكل 2: توزع القطاعات الأربعة في فضاء عناوين المعالج.

يطلق اسم المكدس على جزء من الذاكرة يُخصص لتخزين العناوين والمعطيات أثناء تنفيذ البرامج الفرعية. ويُستخدم سجل قطاع المكدس لتخزين الجزء العلوي (ذي 16 خانة) من عنوان بداية المكدس. وسيرد شرح مفصل لاستخدام المكدس في الفصل الثالث.

3-1-3 مؤشر التعليمات

كما ذكرنا آنفاً، يخزن سجل قطاع البرنامج CS الجزء العلوي من عنوان بداية القطاع الذي ستجلب منه وحدة التواجه التعليمات. يفيد مؤشر التعليمات IP (Instruction Pointer) في تخزين 16 خانة من العنوان الذي ستجلب منه وحدة التواجه التعليمة المقبلة، وهذا العنوان ينتمي حتماً إلى قطاع البرنامج. وبكلمة أخرى، فهذا العنوان يمثل الانزياح الواجب إضافته إلى محتوى سجل CS للوصول إلى التعليمة. فالسجل CS يؤشر نحو القاعدة Base، أي بداية قطاع البرنامج، في حين يخزن مؤشر التعليمات انزياحَ Offset التعليمة المقبلة عن عنوان القاعدة. ويظهر الشكل 3 آلية جمع الانزياح المخزّن في السجل IP إلى العنوان القاعدي CS.



الشكل 3: إضافة محتوى السجل IP إلى السجل CS لتوليد العنوان الحقيقي للتعليمة المقبلة.
أ) المخطط. ب) آلية الحساب.

نلاحظ أن محتوى السجل CS يزاح أولاً بأربع خانات نحو اليسار، ثم يجمع جمعاً مباشراً إلى محتوى السجل IP. فإذا كان محتوى السجل CS هو 348Ah فعند إزاحته يساراً يصبح 348A0h وهذا يمثل عنوان بداية القطاع. يضاف إلى ذلك محتوى السجل IP وهو 4214h، لنحصل على العنوان الحقيقي الكامل 38AB4h. وتُمثل عادةً العناوين الحقيقية في المعالج 8086،المرمزة على 20 خانة، بقيمتين هما: القاعدة والانزياح ويكتبان كما يلي: الانزياح:القاعدة . فمثلاً، يكتب العنوان الحقيقي 38AB4h كما يلي: 348A:4214h.

2-3 وحدة التنفيذ

تخبر وحدة التنفيذ في المعالج 8086 وحدة التواجه عن مكان جلب التعليمات أو المعطيات، وتقوم بتفكيك التعليمات وتنفيذها. وهي تحتوي على الأجزاء الوظيفية التالية:

1-2-3 دارات التفكيك

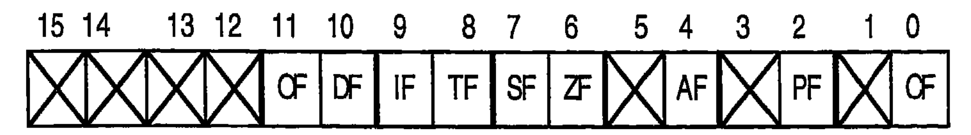
تقوم دارات التفكيك Decoding بترجمة التعليمات المقروءة من الذاكرة إلى سلسلة من الأفعال التي تنفذها وحدة التنفيذ بواسطة وحدة الحساب والمنطق.

2-2-3 وحدة الحساب والمنطق

توكل إلى هذه الوحدة مهمة تنفيذ العمليات الحسابية الأساسية، مثل الجمع والطرح، والعمليات المنطقية البسيطة مثل عمليات AND و OR والإزاحة. وهي تنفذ هذه العمليات على حدين، كل منهما مرمّز على 16 خانة (انظر الشكل 1). وللدلالة على نتائج عملها، تتصل هذه الوحدة بسجل خاص يسمى سجل الرايات أو سجل الحالة.

3-2-3 سجل الرايات

يطلق اسم راية Flag على الخانة التي تدل على تحقق شرط معين من جراء تنفيذ تعليمة ما، أو يمثل إيعازاً محدداً لوحدة التنفيذ. وتحتوي وحدة التنفيذ في المعالج 8086 على سجل رايات طوله 16 خانة، منها 9 خانات ذات دلالة وسبع غير مستخدمة (انظر الشكل 4).



الشكل 4: سجل الرايات في المعالج 8086.

تُستخدم ست رايات في الدلالة على تحقق شرط ما بعد تنفيذ التعليمة الحالية.فعلى سبيل المثال، تدل راية الحمل Carry، عندما تكون مساوية للواحد، على أن نتيجة جمع عددين مرمزين على 16 خانة أكبر من أن تُرمز على العدد نفسه من الخانات. وإذا كان ناتج الجمع عدداً قابلاً للترميز على 16 خانة فإن الراية ستبقى مساوية للصفر. ونصف فيما يلي عمل رايات المعالج:

- راية الحمل CF (Carry Flag): وقد ذُكرت سابقاً.
- راية التثبت PF (Parity Flag): وتصبح مساوية للواحد عندما يكون عدد الخانات الواحدية في القيمة الناتجة من وحدة الحساب والمنطق زوجياً.
- راية الحمل المساعد AF (Auxiliary Flag): وتصبح مساوية للواحد عندما يكون ناتج جمع الأوزان الدنيا للعددين أكبر من أن يُرمز على 8 خانات.
- راية الإشارة SF (Sign Flag): وتصبح مساوية للواحد عندما يكون حاصل طرح عددين سالباً.
- راية الفائض OF (Overflow Flag): وتصبح مساوية للواحد عندما يكون جداء عددين أكبر من أن يُرمز على 16 خانة.

أما الرايات الثلاث الباقية، وتسمى رايات التحكم، فهي تُستخدم

في قيادة بعض العمليات في المعالج، ويختلف عملها عن الرايات السابقة من حيث طريقة الكتابة فيها. فقيم الرايات الست الأولى تكتبها وحدة التنفيذ، أما رايات التحكم فيمكن للمبرمج أن يكتب فيها القيمة المناسبة. وهذه الرايات هي:

- راية التنفيذ الخطوي TF (Trap Flag): وتسمح، عندما تصبح قيمتها مساوية للواحد، بتنفيذ البرنامج خطوة فخطوة.
- راية المقاطعة IF (Interrupt Flag): وهي تفيد في سماح أو منع المقاطعات في المعالج.
- راية الاتجاه DF (Direction Flag): وهي تُستخدم أثناء تنفيذ التعليمات الخاصة بسلاسل المحارف.

4-2-3 السجلات العامة الاستخدام

نلاحظ من الشكل 2 أن لوحدة التنفيذ ثمانية سجلات عامة تسمى AL, AH, BL, BH, CL, CH, DL, DH.

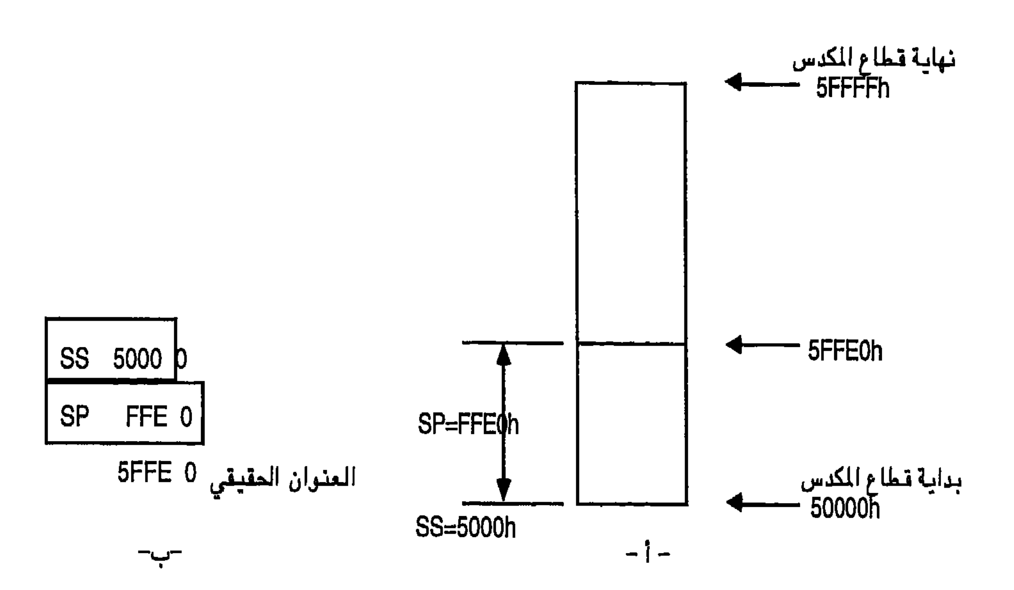
يمكن استخدام هذه السجلات استخداماً منفصلاً لتخزين المعطيات الثانوية المرمزة على 8 خانات. ويُطلق اسم المراكم Accumulator على السجل AL لأنه يتمتع بصفات خاصة.

من جهة أخرى، يمكن دمج كل سجلين معاً لتخزين كلمات مرمّزة على 16 خانة، فيُدمج معاً السجلان AL و AH والسجلان BL وBH والسجلان CL و CH والسجلان DL وDH. ويُرمز إلى كل من هذه السجلات المدمجة باسم جديد. فالزوج AL-AH يسمى AX والزوج BH-BL يسمى BX، وكذلك يسمى السجلان CL-CH السجل CX، وأخيراً يُسمى DL-DH السجل DX. ويقوم السجل AX بدور المراكم في العمليات على الأرقام المرمّزة على 16 خانة.

5-2-3 سجل مؤشر المكدس

كما ورد سابقاً، يُطلق اسم المكدس Stack على جزء الذاكرة

المخصص لتخزين المعطيات والعناوين اللازمة لتنفيذ البرامج الفرعية. ويتيح المعالج 8086 تعريف قطاع من الذاكرة يصل إلى 64K ثمانيّة للقيام بدور المكدس. يُخزن الجزء العلوي من عنوان بداية القطاع في السجل SS، أما انزياح العنوان الحالي عن عنوان البداية فيُخزن في السجل SP. ويُقصد بالعنوان الحالي عنوان موقع الذاكرة الذي خُزنت فيه آخر كلمة في المكدس. ويسمى ذلك الموقع بأعلى المكدس أو قمة المكدس Top of Stack. ونجد في الشكل 5 مثالاً عن عمل المكدس.



الشكل 5: العنونة في قطاع المكدس.
أ) المخطط. ب) آلية الحساب.

6-2-3 سجلات الدليل

تحوي وحدة التنفيذ، إضافة إلى مؤشر المكدس، ثلاثة سجلات طول كل منها 16 خانة، وهي: مؤشر القاعدة BP (Base Pointer)، ودليل المصدر SI (Source Index)، ودليل الوجهة DI (Destination Index). يمكن

استخدام هذه السجلات في تخزين المعطيات الثانوية كأي سجل عام. إلا أن استخدامها الرئيسي هو تخزين انزياح معلومة ما في أحد قطاعات الذاكرة. فعلى سبيل المثال، يستخدم السجل SI لتخزين انزياح كلمة في قطاع المعطيات، وهذا يعني أن عنوان تلك الكلمة الحقيقي ينتج من إزاحة محتوى السجل DS أربع خانات نحو اليسار، وإضافة محتوى السجل SI إلى النتيجة.

4 إشارات المسرى

ذكرنا سابقاً أن المعالج هو وحدة تقوم بتنفيذ التعليمات المخزّنة في الذاكرة. ولكي يستطيع المعالج تنفيذ تعليمة ما ينبغي أن يقرأ هذه التعليمة من الذاكرة.

بعد التحليل، يقوم المعالج بتنفيذ التعليمة (سواء تضمنت التعليمة عملية حسابية أو منطقية). وقد يضطر المعالج، تبعاً لنوع التعليمة، إلى تخزين النتيجة في الذاكرة.

بمعنى آخر، لابد للمعالج من القيام بعمليات قراءة وكتابة في الذاكرة أو في إحدى الطرفيات الأخرى. ويستطيع ذلك بواسطة الإشارات الإلكترونية [1] التي يتبادلها مع الذاكرة أو الطرفيات.

نصف في هذه الفقرة تسلسل الإشارات المتبادلة على مسرى المعالج أثناء عمليتي القراءة والكتابة.

4-1 عملية القراءة من الذاكرة

عندما يقرأ المعالج قيمة ما من الذاكرة فإنه يستخدم خطوط العنونة والمعطيات AD_{0-15} وخطوط العنونة العليا A_{16-19} وخطوط التحكم التالية: $\overline{RD}$, ALE, $M/\overline{IO}$, S0-S2, READY, $\overline{DEN}$, $DT/\overline{R}$ (انظر الشكل 6).

يتضمن تنفيذ تعليمة ما في المعالج عدة عمليات أساسية (وهي

1 انظر في الملحق الثالث وصف مرابط المعالج 8086 الخارجية.

جلب التعليمة وتحليلها وتنفيذها وتخزين النتائج). تحتاج كل عملية أساسية إلى عدد من أدوار الساعة CLK للتنفيذ، وتسمى كل عملية منها دور الآلة Machine Cycle. كما يطلق اسم دور التعليمة Instruction Cycle على مجموعة أدوار الساعة CLK اللازمة لتنفيذ تعليمة ما تنفيذاً كاملاً. فيحتوي إذن دور التعليمة على دور أو أكثر من أدوار الآلة، وهذا يحتوي بدوره على مجموعة أدوار من الساعة CLK.

أثناء القراءة، يؤهل المعالج أولاً الإشارة $M/\overline{IO}$ فيضعها علي المستوى العالي (5 volts) إذا أراد مخاطبة الذاكرة، أو على المستوى المنخفض (0 volt) في حال مخاطبة معبر د/خ. (يشار على المخطط بتقاطع عندما تتغير قيمة الإشارة.)

ولتحليل المخطط ينبغي تتبع الإشارات من اليسار نحو اليمين.

بعد وضع قيمة $M/\overline{IO}$ المناسبة، يضع المعالج عنوان الذاكرة (أو المعبر) المراد قراءته مستخدماً المرابط AD_{0-15} و A_{16-19}. وفي الوقت ذاته، يولِّد نبضة على الخرج ALE للدلالة على جاهزية العنوان الموجود على الخطوط AD_{0-15}. كما يحدد المعالج طول الكلمة المطلوبة بواسطة الإشارة $\overline{BHE}$.

بعد وضع العناوين، يستعد المعالج لقراءة القيمة من الذاكرة (أو من المعبر) فيؤهل إشارة القراءة $\overline{RD}$، وهذا ما يدعو الطرفية أو الذاكرة إلى وضع المعطيات على الخطوط.

أما إشارة الجاهزية READY فهي تدل على أن الذاكرة أو الطرفية جاهزة لتبادل المعلومات مع المعالج. فإذا وجد المعالج على هذا الدخل القيمة 0 (أي إن إشارة الجاهزية غير فعّالة) فإنه سيدخل في حالة انتظار Wait State إلى أن تصبح الطرفية جاهزة، وعندئذٍ يستأنف عمله.

هناك إشارتان يخرجهما المعالج لقيادة دارات مساعدة خارجية وهما $DT/\overline{R}$, $\overline{DEN}$. تدل الإشارة الأولى $DT/\overline{R}$ على اتجاه المعطيات.

فمثلاً في حالة القراءة، تأخذ هذه الإشارة القيمة '0' للدلالة على أن المعلومات تتجه من الذاكرة نحو المعالج. أما الإشارة $\overline{DEN}$، فهي تتأهل عندما تظهر المعطيات على خطوط العنونة والمعطيات AD_{0-15}.

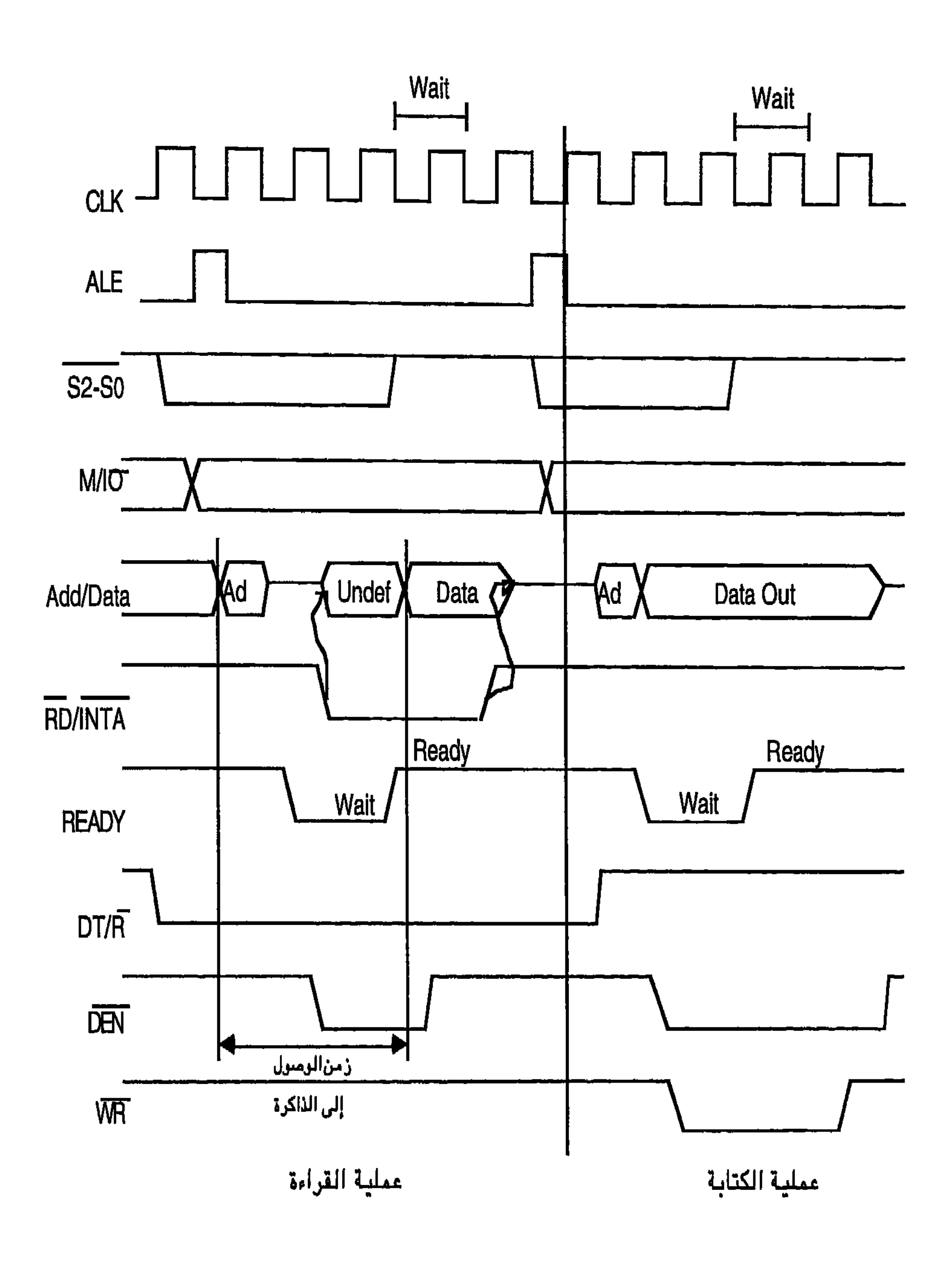
2–4 عملية الكتابة في الذاكرة

يظهر الشكل 6 أيضاً مخططاً لإشارات المسرى أثناء عملية الكتابة. ويبدو، بالمقارنة بحالة القراءة، أن هناك تشابهاً واضحاً في تسلسل العمل.

يضع المعالج أولاً القيمة المناسبة على الخرج $M/\overline{IO}$ لتحديد هوية الوجهة (ذاكرة أم معبر)، ثم يخرج عنوان الموقع المطلوب الكتابة فيه على خطوط العنونة والمعطيات AD_{0-15} وA_{16-19} ، ويؤهل الإشارة $\overline{BHE}$ عند تعامله مع كلمة مرمزة على 16 خانة. بعد ذلك، يولّد نبضة على الخرج ALE للدلالة على جاهزية العناوين على المسرى.

أما المرحلة التالية، فهي مختلفة عن حالة القراءة. إذ يخرج المعالج القيمة المراد كتابتها على الخطوط AD_{0-15}، ويؤهل إشارة الكتابة $\overline{WR}$ للإيعاز إلى الذاكرة أو إلى معبر الدخل/الخرج بالكتابة. يدخل بعدئذٍ المعالج في حالة الانتظار لإشارة الجاهزية READY، وحين وصولها ينتقل إلى تنفيذ التعليمة التالية.

يضع المعالج، أثناء عملية الكتابة، القيمة المناسبة على الخط $DT/\overline{R}$ لتحديد اتجاه المعلومات. ففي هذه الحالة، تكون قيمة الإشارة '1'، لأن المعلومات تتجه من المعالج نحو الذاكرة أو الطرفية. ويؤهل الخط $\overline{DEN}$ فور خروج المعطيات على خطوط المسرى لتنبيه الدارات الخارجية إلى وجود المعطيات.



الشكل 6: إشارات المسرى أثناء عمليتي القراءة والكتابة.

الفصل الثاني

الوحدات المحيطية والدخل/الخرج

1 مقدمة

ذكرنا في الفصل السابق إن معظم الوظائف الحسابية الرئيسية للحواسيب تحتاج إلى مكونين أساسيين هما وحدة المعالجة المركزية والذاكرة، إذ تجلب وحدة المعالجة المركزية التعليمات والمعطيات من الذاكرة وتعود لتخزن النتائج فيها. ولكن الحاسوب يحتاج إلى عناصر أخرى يمكن تسميتها بنظام الدخل/الخرج، الغرض الأساسي منها نقل المعطيات بين الذاكرة أو وحدة المعالجة المركزية من جهة، وبين العالم الخارجي (الذي يتضمن الوحدات المحيطية، كالطابعات مثلاً، ووحدات التحكم الخاصة بهذه التجهيزات) من جهة أخرى.

يمكن تسمية عمليات الإدخال والإخراج، التي تقوم وحدة المعالجة المركزية بالتحكم التام فيها بعمليات الدخل/الخرج المبرمجة Programmed I/O، فتنفذ وحدة المعالجة المركزية البرنامج الذي يقوم بإطلاق وتوجيه وإنهاء عمليات الدخل/الخرج. لاتحتاج عمليات الدخل/الخرج المبرمجة إلى تجهيزات وعناصر خاصة، ولكنها تأخذ الكثير من وقت وحدة المعالجة المركزية في عمليات تحكم بسيطة، كالتحقق من وضعية أحد تجهيزات الدخل/الخرج لمعرفة حاجة الجهاز إلى تخديم من وحدة المعالجة المركزية.

ويمكن أيضاً القيام بنقل كتلة من المعطيات من وإلى الذاكرة من

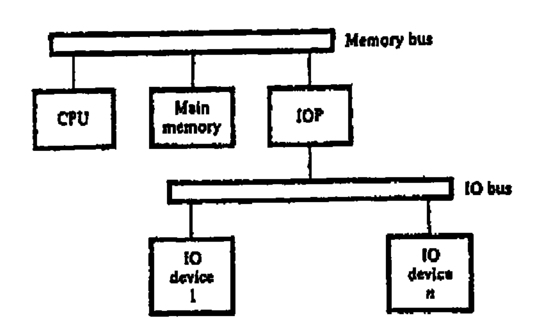
خلال جهاز الدخل/الخرج، دون تدخل وحدة المعالجة المركزية، وذلك بزيادة تعقيد بنية التحكم في الدخل/الخرج بدرجة طفيفة نسبياً. ويتطلب ذلك أن يكون جهاز الدخل/الخرج، أو المتحكم الخاص به، قادراً على توليد عناوين الذاكرة ونقل المعلومات من أو إلى الذاكرة، إضافةً إلى إمكان طلب استحواذ المسرى Bus وآلية انتخاب. تبقى وحدة المعالجة المركزية في هذا النمط من العمل مسؤولة عن إطلاق عملية نقل كل كتلة معطيات، ومن ثم يقوم جهاز الدخل/الخرج بإجراء عملية النقل دون حاجة وحدة المعالجة المركزية إلى تنفيذ أي برنامج، وتتفاعل وحدة المعالجة المركزية مع جهاز الدخل/الخرج عندما يطلب هذا الجهاز الاستحواذ على المسرى من وحدة المعالجة المركزية. نسمي هذا النمط من الدخل/الخرج بالنفاذ المباشر إلى الذاكرة DMA (Direct Memory Access).

كذلك قد تُزَوَّد أجهزة الدخل/الخرج بدارات تمكنها من طلب الخدمة من وحدة المعالجة المركزية، أي طلب تنفيذ برنامج محدد لتخديم جهاز الدخل/الخرج، يسمى هذا الطلب بالمقاطعة Interrupt. ويحرر هذا الإمكان وحدة المعالجة المركزية من عبء اختبار حالة الدخل/الخرج دورياً. تسبب المقاطعة انتقال وحدة المعالجة المركزية إلى تنفيذ برنامج معالجة المقاطعة، بعد حفظ حالة البرنامج الذي كانت تقوم بتنفيذه عند ورود طلب المقاطعة، وتعود إلى متابعة تنفيذ البرنامج المُقاطَع، عند الانتهاء من خدمة المقاطعة [1]. تملك معظم الحواسيب اليوم إمكانات النفاذ المباشر إلى الذاكرة والمقاطعة، وهذا ما يتطلب إضافة وحدات تحكم خاصة بالمقاطعة وبالنفاذ المباشر إلى الذاكرة.

توجد كذلك اليوم وحدات تحكم تمنح جهاز الدخل/الخرج إمكان التحكم المطلق في عمليات الدخل/الخرج، وتسمى معالجات الدخل/الخرج I/O Processors (IOP)، وهي تملك إمكان النفاذ المباشر إلى الذاكرة ومقاطعة وحدة المعالجة المركزية، إضافةً إلى إمكان تنفيذ

1 انظر الفصل الرابع، الفقرة 4 من أجل دراسة المقاطعة من الناحية البرمجية.

برامج خاصة بعمليات الدخل/الخرج. تستطيع معالجات الدخل/الخرج القيام بعدة عمليات نقل معطيات منفصلة بين الذاكرة وجهاز (أو عدة أجهزة) في الدخل/الخرج دون الرجوع إلى وحدة المعالجة المركزية، وغالباً ما تتصل بالجهاز الذي تتحكم فيه عن طريق مسرى منفصل عن المسرى الأساسي يسمى بمسرى الدخل/الخرج I/O BUS. كما يبين الشكل 1.



الشكل 1: استخدام مسريين منفصلين
أحدهما للذاكرة الأساسية والثاني للدخل/الخرج.

2 عمليات الدخل/الخرج المبرمجة

سنبحث أولاً في طريقة التحكم في الدخل/الخرج على وجه مبرمج، وهي طريقة متاحة في معظم الحواسيب، وتحتاج كما ذكرنا آنفاً إلى تنفيذ كافة عمليات الدخل/الخرج تحت السيطرة المباشرة لوحدة المعالجة المركزية، أي إن كل عملية نقل معطيات من جهاز الدخل/الخرج تحتاج إلى تنفيذ تعليمة من قبل المعالج. يجري عادة النقل بين سجل في وحدة المعالجة المركزية (كسجل المراكم الرئيسي في وحدة المعالجة المركزية مثلاً) وبين سجل مؤقت Buffer Register متصل بجهاز الدخل/الخرج. لا يملك جهاز الدخل/الخرج إمكان النفاذ المباشر إلى الذاكرة الأساسية، وتحتاج عملية نقل المعطيات من جهاز

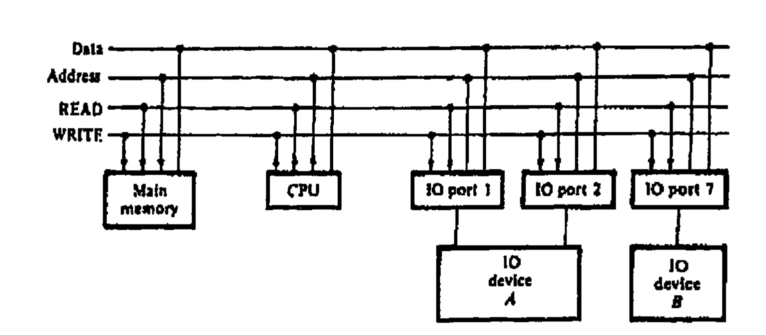
الدخل/الخرج إلى الذاكرة الأساسية إلى تنفيذ تعليمتين من قبل وحدة المعالجة المركزية هما: تعليمة إدخال لنقل الكلمة من جهاز الدخل/الخرج إلى وحدة المعالجة المركزية، وتعليمة تخزين لنقل الكلمة من وحدة المعالجة المركزية إلى الذاكرة الأساسية.

1-2 عنونة الدخل/الخرج

تتخاطب كل من أجهزة الدخل/الخرج والذاكرة الأساسية ووحدة المعالجة المركزية، بواسطة مسرى وحيد مشترك تستخدم فيه نفس خطوط العنونة التي تنتخب مواقع الذاكرة الرئيسية لاختيار أجهزة الدخل/الخرج. تسمى كل عقدة وصل بين المسرى الأساسي وجهاز الدخل/الخرج معبر دخل/خرج I/O Port، ويسند إليها عنوان فريد. قد يتضمن معبر الدخل/الخرج سجل تخزين مؤقت للمعطيات، وهذا ما يجعله يبدو مشابهاً للذاكرة الأساسية من وجهة نظر وحدة المعالجة المركزية.

يخصص في بعض الحواسيب جزء من مساحة العنونة الخاصة بالذاكرة الأساسية لمعابر الدخل/الخرج، وتسمى هذه الاستراتيجية بالدخل/الخرج المحجوز من الذاكرة Memory-Mapped I/O. في هذه الحالة، تصبح تعليمات جلب المعطيات من (أو تخزينها في) الموقع X عمليات إدخال وإخراج من وإلى الموقع X، الذي يمثل عنواناً لمعبر الدخل/الخرج. وتستخدم تعليمات الجلب Fetch والتخزين Store من/في الذاكرة الاعتيادية لنقل كلمة معطيات من وإلى جهاز الدخل/الخرج دون الحاجة لتعليمات خاصة بالدخل/الخرج. يبين الشكل 2 البنية الضرورية لهذا النمط من الدخل/الخرج.

تستخدم خطوط التحكم في القراءة Read والكتابة Write، التي تقوم بتفعيلها وحدة المعالجة المركزية عند تفكيك ترميز التعليمات التي تتعامل مع الذاكرة، بإطلاق إما دورة نفاذ إلى الذاكرة أو عملية نقل من معبر دخل/خرج.



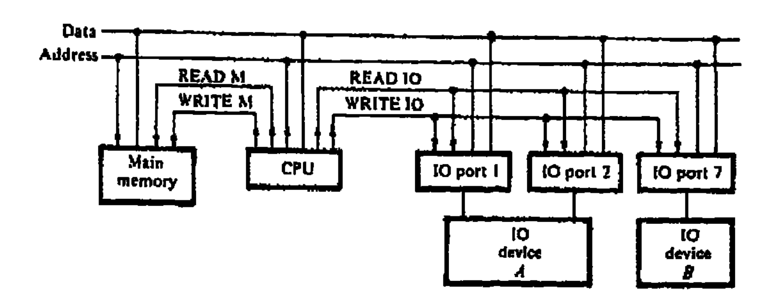
الشكل 2: الدخل/الخرج المبرمج على مساحة عنونة مشتركة بين الذاكرة والدخل/الخرج (دخل/خرج محجوز من الذاكرة).

هناك بنية أخرى شائعة الاستخدام تكون فيها مساحات عنونة الذاكرة مفصولة عن مساحات عنونة الدخل/الخرج، كما يبين الشكل 3، حيث تقوم تعليمات التعامل مع الذاكرة بتشغيل خطوط التحكم في القراءة والكتابة الخاصة بالذاكرة ReadM و WriteM دون أن تؤثر في أجهزة الدخل/الخرج. نحتاج في هذه البنية إلى تعليمات دخل/خرج خاصة لتقوم بتشغيل خطوط التحكم في القراءة والكتابة الخاصة بالدخل/الخرج ReadI/O و WriteI/O التي تقوم بنقل كلمة معطيات بين معبر الدخل/الخرج المعنونة ووحدة المعالجة المركزية. في هذه الاستراتيجية يمكن جهازَ الدخل/الخرج وموقعَ ذاكرة رئيسي أن يكون لهما نفس العنوان؛ وتستخدم عائلة المعالجات 8080 من شركة إنتل INTEL، على سبيل المثال، هذه البنية للدخل/الخرج.

2-2 تعليمات الدخل/الخرج الأساسية

يمكن تحقيق الدخل/الخرج المبرمج بتعليمتين برمجيتين فقط. تنقل مثلاً التعليمة IN X كلمة من معبر الدخل/الخرج X إلى المراكم Accumulator في المعالج، في حين تنقل تعليمة OUT X كلمة من المراكم إلى معبر الدخل/الخرج X. يتوقع المعالج عند تنفيذه لتعليمات

الدخل/الخرج IN/OUT أن تكون معبر الدخل/الخرج المعنونة جاهزة للاستجابة للتعليمة المنفذة.



الشكل 3: الدخل/الخرج المبرمج بمساحات عنونة للدخل/الخرج منفصلة عن مساحة عنونة الذاكرة.

عند عدم استخدام المصافحة Handshaking، أي عندما لا يقوم جهاز الدخل/الخرج بتوليد إشارة إشعار Acknowledgement، يجب أن يقوم جهاز الدخل/الخرج بنقل المعطيات من وإلى المسرى خلال مدة محددة. إذن لضمان عدم ضياع المعطيات، وحتى لا يمتد زمن تعليمات الدخل/الخرج كثيراً، يفضل أن تتعرف وحدة المعالجة المركزية حالة جهاز الدخل/الخرج لكي تجري عملية النقل عندما يكون جهازالدخل/الخرج في حالة الجاهزية. في هذا النمط من عمليات الدخل/الخرج تبرمج وحدة المعالجة المركزية لاختبار حالة جهاز الدخل/الخرج قبل إطلاق عملية نقل المعطيات. يمكن غالباً تمثيل حالة الجهاز بخانة اثنانية وحيدة تتصل بأحد خطوط المعطيات.

يتطلب تحديد حالة جهاز الدخل/الخرج الحلقة البرمجية التالية:

1- Read the status information.
2- Test the status to determine if device is ready to begin data transfer.
3- If not ready, return to step 1; otherwise proceed with data transfer.

يبين المقطع التالي برنامجاً، مكتوباً بلغة المجمّع Assembler الخاصة بالمعالج 8080، يقوم بنقل كلمة معطيات من جهاز الدخل/الخرج إلى مراكم وحدة المعالجة المركزية. يفترض البرنامج أن

جهاز الدخل/الخرج متصل بالبوابتين 1 و 2 (الجهاز A في الشكل 2) وأن حالة جهاز الدخل/الخرج موجودة دوماً على المعبر 1 وأن المعطيات المطلوبة موجودة على المعبر 2 عندما تدل كلمة الحالة على الجاهزية.

```
Wait:   IN 1          ; read I/O device status in to accumnlator.
        CPI READY     ; compare immediate word READY
                      ; to accumulator:
                      ; If equal set flag Z=1,
                      ; Otherwise set flag Z=0.
        JNZ Wait      ; If Z # 1 (I/O device not ready) jmup to wait.
        IN 2          ; read data word in to accumulator
```

3-2 تعليمات دخل/خرج إضافية

إذا كانت الطريقة الأساسية للتحكم في الدخل/الخرج هي طريقة الدخل/الخرج المبرمج، فقد توفر وحدة المعالجة المركزية بعض التعليمات الإضافية، إلى جانب تعليمتي IN و OUT المذكورتين سالفاً، كتعليمات نقل كتلة من المعطيات، الخ...

يمكن تبسيط عملية وصل أجهزة الدخل/الخرج إلى نظم الحواسيب باستخدام دارات إلكترونية جاهزة معيارية تسمى بدارات الترابط للدخل/الخرج أو معابر الدخل/الخرج أو موائمات (واجهات) الترابط. تسمح هذه الدارات بوصل أجهزة مختلفة المواصفات إلى مسرى مشترك باستخدام أقل عدد ممكن من العناصر الإلكترونية.

ويعد السجل المؤقت الوحيد الكلمة أبسط دارة ترابط. وهو يعمل كعلبة بريد أثناء عمليات الدخل/الخرج، فيسند إليه عنوان فريد ويجري النفاذ إليه تماماً كموقع في الذاكرة الأساسية. وتفيد هذه الدارة في النقل التفرعي للمعطيات (أي كلمة فكلمة).

ثمة نمط آخر من دارات الترابط تسمى بالمرسِلات والمستقبِلات اللامتزامنة العامة Universal Asynchronous Receivers-Transmitters (نختصرها بلفظة UARTs) التي تسمح بربط الحواسيب بأجهزة الدخل/الخرج ربطاً تسلسلياً (خانة اثنانية تلو الأخرى) كما هو الحال عند ربط الحاسوب بخط الهاتف. تتكون المرسلات والمستقبلات

اللامتزامنة العامة أساساً من مسجل إزاحة Shift Register يحول تتالي المعطيات التسلسلية إلى تفرعية والعكس بالعكس.

3 دارات الدخل/الخرج المبرمجة وعمليات الدخل/الخرج المحكومة بالمصافحة

تحتوي معظم دارات الدخل/الخرج، كالدارة 8255A، معبرين أو ثلاثة يمكن برمجتها لتعمل بأحد الأنماط المتاحة، التي يسمح كل منها باستخدام المعبر ليقوم بعمليات دخلٍ/خرجٍ تفرعية وفق إحدى الطرق الشائعة التي سنستعرضها فيما يلي.

3-1 طرق نقل المعطيات

3-1-1 عمليات الدخل/الخرج البسيطة

عندما نحتاج إلى قراءة حالة دخل/خرج رقمي بسيط (مفتاح أو قاطع حراري Thermostat مثلاً) من معالج صغري، نصلُ هذا الدخل إلى حد خطوط معبر الدخل/الخرج ونقرأ المعبر. تكون المعطيات موجودة و جاهزة للقراءة في أي وقت. كذلك الأمر عند الحاجة إلى إخراج معطيات رقمية لوحدة إظهار بسيطة، كثنائي مُصْدرٍ للضوء Light Emitting Diode (LED)، فنصل دخل سجل العزل، المتصل بالثنائي، إلى أحد خطوط معبر خرج، ونضع عليه برمجياً المستوى المنطقي المناسب لإضاءة الثنائي الضوئي. ويكون الثنائي موجوداً دوماً وجاهزاً لاستقبال الإشارة.

3-1-2 عمليات الدخل/الخرج بالقدح

تكون المعطيات متوفرة على جهاز الدخل أو الخرج في كثير من التطبيقات خلال مدة معينة محدودة يجب قراءتها خلالها. كمثال على

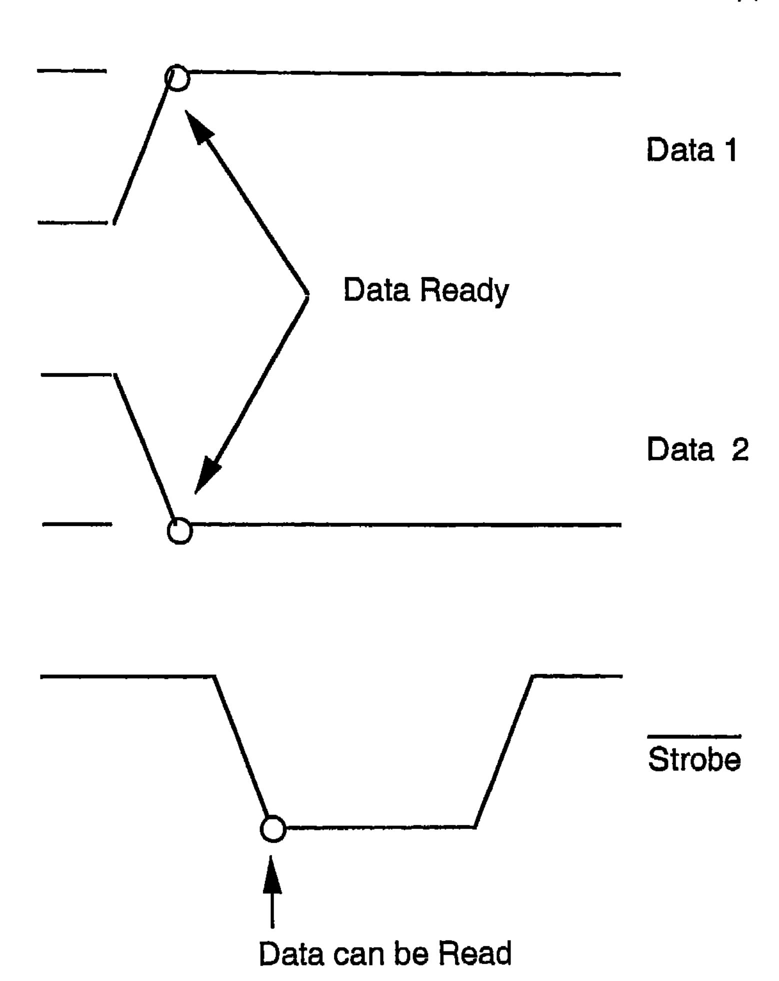
هذه العمليات نذكر مثلاً عملية القراءة من لوحة مفاتيح لحظية Keypad لأحد الأحرف المرمزة وفق ترميز ASCII. عند الضغط على أحد المفاتيح تضع دارات لوحة المفاتيح ترميز ASCII للمفتاح المضغوط على ثمانية خطوط على التفرع، ثُم ترسل هذه الدارات إشارة قدح على خط آخر للدلالة أن المعطيات المتوفرة على الخطوط الثمانية جاهزة. يمكن وصل خط القدح إلى خط معبر دخل وتقصيه polling باستمرار لتحري معطيات جديدة متوفرة للإدخال؛ يمكن كذلك وصل خط القدح إلى أحد خطوط طلب المقاطعة للمعالج الصغري، بحيث يقوم برنامج خدمة المقاطعة بقراءة المعطيات عن حدوث طلب المقاطعة. في هذه الحالة يعتمد نقل المعطيات على الزمن، ويمكن قراءة المعطيات عندما تُعلمنا إشارة القدح بأن المعطيات متوفرة وصالحة. يبين الشكل 4 هذا النمط من عمليات الدخل/الخرج.

تصلح هذه الطريقة لنقل المعطيات بمعدل بطيء (كالإدخال من لوحات المفاتيح لحاسوب أو معالج صغري)، ولكنها لاتصلح لنقل المعطيات بمعدل سريع، بسبب عدم وجود إشارة تُعلم الجهاز، الذي يقوم بإرسال المعطيات، متى يمكنه إرسال الكلمة التالية من المعطيات. أي قد يقوم الجهاز المرسل بإرسال كلمات المعطيات بمعدل أسرع من قدرة الجهاز المستقبل على القراءة، ونحتاج للقضاء على هذه المشكلة إلى تبادل المعطيات وفق مراسيم المصافحة للنقل .Hand Shake

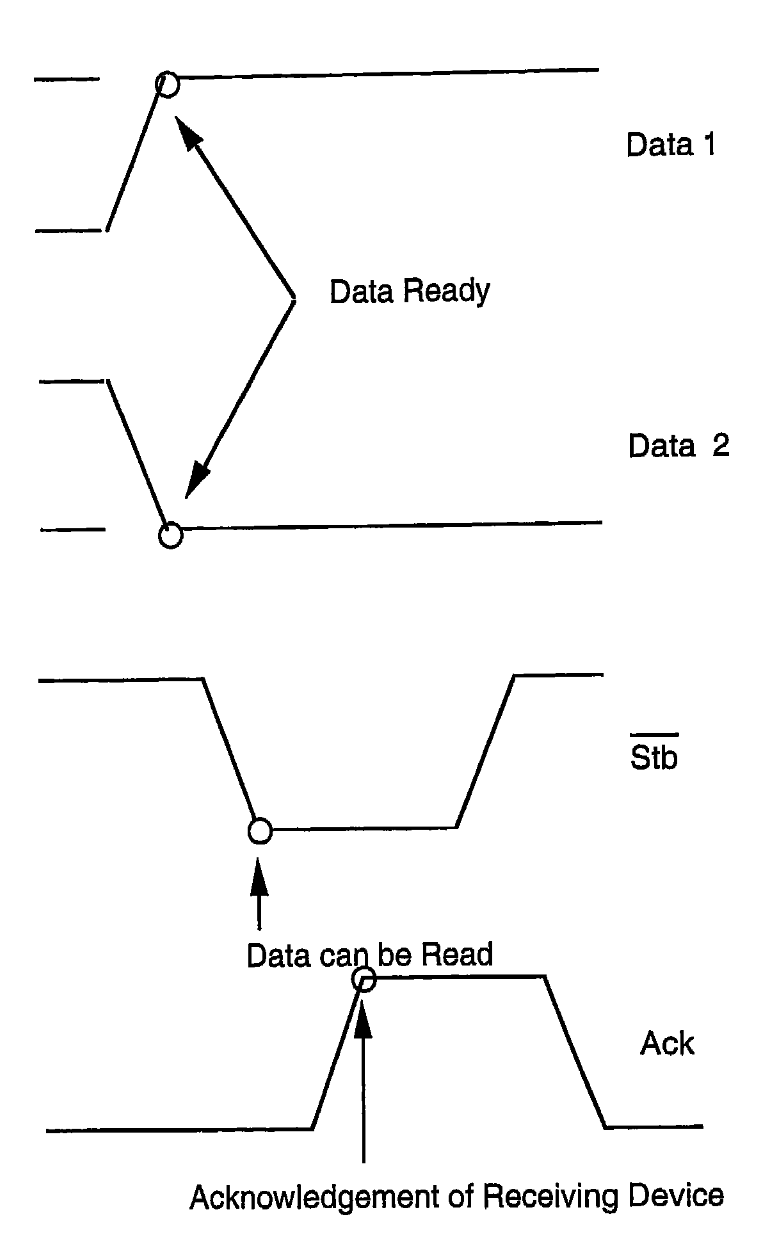
3-1-3 عمليات الدخل/الخرج بالمصافحة الوحيدة

في هذا النمط من العمليات، يضع الجهاز المحيطي المعطيات التفرعية على مخارجه ويرسل إشارة قدح للمعالج الصغري، الذي يكشف هذه الإشارة (إما بالتقصي أو المقاطعة) ويقرأ المعطيات. ثم يقوم المعالج بإرسال إشارة إشعار إلى الجهاز المحيطي لإعلامه أن بإمكانه إرسال الكلمة التالية من المعطيات. تسمى هذه العملية، الموضحة بالشكل 5، بالمصافحة، ولا يمكن للمرسل إرسال المعطيات

قبل أن يعلن الجهاز المستقبل جاهزيته لاستقبال كلمة المعطيات التالية بإشارة ACK.



الشكل 4: الدخل/الخرج المحكوم بإشارة قدح.



الشكل 5: الدخل/الخرج بالمصافحة الوحيدة.

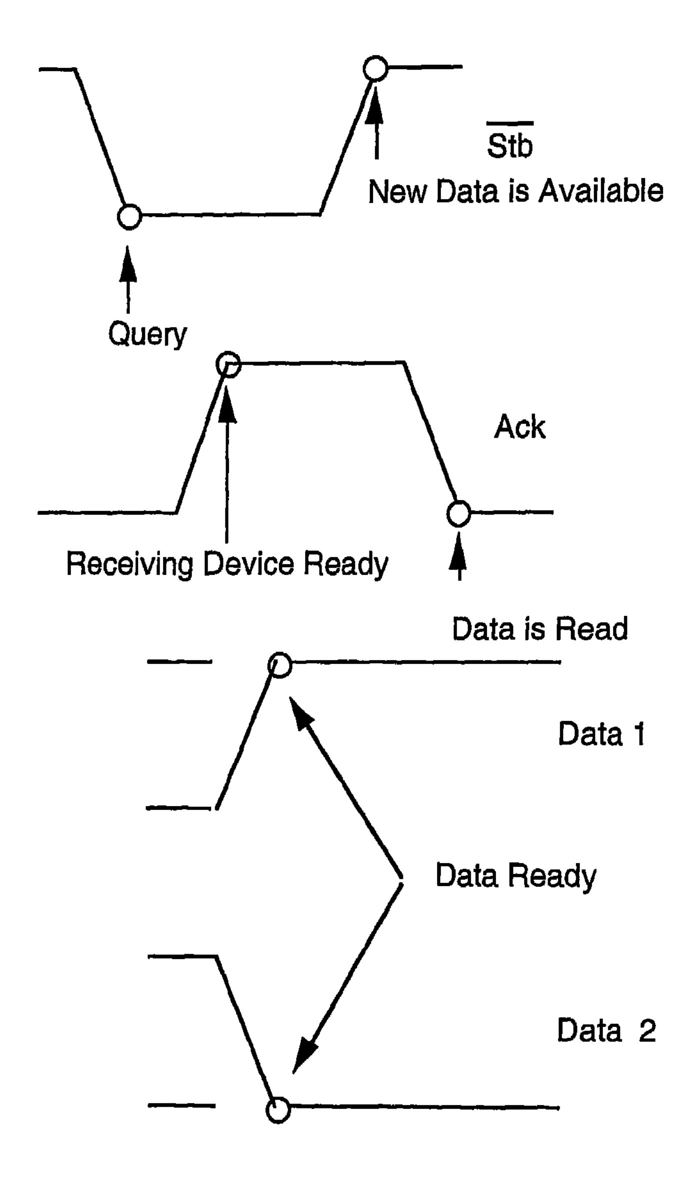
4-1-3 عمليات الدخل/الخرج بالمصافحة المزدوجة

تستخدم المصافحة المزدوجة، الموضحة بالشكل 6، عند الحاجة إلى توافق أكبر بين المرسل والمستقبل. في هذا النمط من المصافحة يستعلم الجهاز المرسل عن جاهزية المستقبل بوضع خط القدح على المستوى المنطقي المنخفض. فإذا كان المستقبل جاهزاً، يقوم بإعلام المرسل عن جاهزيته بوضع خط الإشعار على المستوى المنطقي العالي. عندها يضع المرسل المعطيات الجديدة على خطوط المعطيات ويجعل المستوى المنطقي على خط القدح $\overline{STB}$ عالياً ليُعلم المستقبل بأن المعطيات الحديثة متوفرة؛ يقوم المستقبل بقراءة المعطيات ثم يضع منطقاً منخفضاً على خط الإشعار ACK لإعلام المرسل بأن المعطيات قد تمت قراءتها وأن المستقبل ينتظر معطيات جديدة.

وعند عملية إخراج من المعالج الصغري إلى الجهاز المحيطي وفق هذا النمط، تبقى الإشارات السابقة نفسها صالحة للتعبير عن تتابع العمل، ويقوم المعالج الصغري بوضع إشارات القدح والمعطيات على الخطوط.

يمكن كشف إشارة القدح أو الإشعار إما بالتقصي البرمجي أو بالمقاطعة، وكذلك الأمر عند توليد إشارات القدح والإشعار، ولكن يفضل عادة استخدام المقاطعة لأنها، كما سبق وذكرنا، تجعلنا نستفيد من وقت المعالج استفادة أكثر فاعلية.

لهذا السبب، صممت دارات المعابر التفرعية، كالمعبر 8255A، بحيث يمكن برمجتها لتقوم آلياً بإدارة عمليات المصافحة دون تدخل المعالج. فمثلاً، يمكن برمجة المعبر لاستقبال إشارة قدح من جهاز محيطي وإرسال إشارة مقاطعة إلى المعالج، ومن ثم إرسال إشارة إشعار إلى الجهاز المحيطي في الوقت المناسب.



الشكل 6: الدخل/الخرج بالمصافحة المزدوجة.

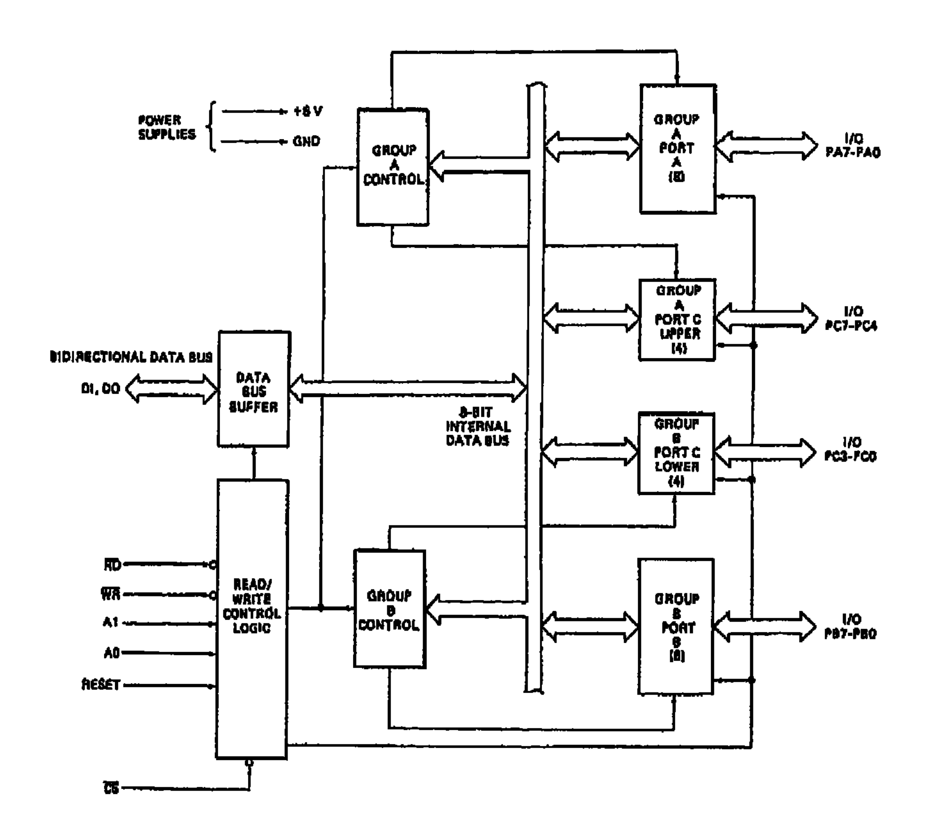
يبين الشكل 7 البنية العامة للدارة 8255A المصممة لربط أجهزة الدخل/الخرج إلى معالجات صغرية من عائلة 8080 من شركة INTEL، تصنع هذه الدارة على رقاقة متكاملة وحيدة وتوضع بغلاف له 40 مربطاً: ثمانية مرابط توصل إلى مسرى المعطيات الثنائي الاتجاه للمعالج الصغري و 24 دخل/خرج يمكن وصلها إلى جهاز دخل/خرج أو أكثر. يمكن تحديد عمل مرابط الدخل/الخرج (دخل أم خرج) عن طريق كلمة تحكم يصدرها المعالج وتختزن في سجل داخل رقاقة المعبر؛ تستخدم كلمة التحكم هذه لانتخاب أحد أنماط عمل الدارة. تقسم مرابط الدارة الأربعة والعشرون إلى مجموعات تتكون كل منها من ثمانية مرابط. نرمز إلى المجموعة الأولى بالمعبر A، وإلى المجموعة الثانية بالمعبر B، وإلى المجموعة الثالثة بالمعبر C، وتعامل كل مجموعة على أنها معبر دخل/خرج مستقل. يقسم المعبر C بدوره إلى مجموعتين تحوي كل منها أربع خانات اثنانية هي CA و CB وغالباً ما تستخدم كخطوط تحكم، مثلاً كخطوط حالة لكل من المعبرين A و B.

يُستخدم خطا العنونة A0 A1 لاختيار إحد المعابر الثلاثة لإجراء عملية الدخل/الخرج، أما العنوان الرابع فيستخدم لكتابة كلمة تحكم في سجل التحكم الداخلي للدارة.

للمعبر A العنوان 00، وللمعبر B العنوان 01، وللمعبر C العنوان 10، على حين يحتل سجل التحكم العنوان 11. نستخدم كلمة التحكم لنحدد عمل المعابر C, B, A كمعابر دخل، أو خرج، أو كمعابر دخل/خرج ثنائية الاتجاه (وهذا متاح للمعبرين A و B فقط). كذلك يمكن برمجة بعض خطوط المعبر C لتقوم بتوليد إشارات المصافحة والمقاطعة آلياً، استجابةً لورود تراكيب معينة للدخل.

يؤهل المدخل $\overline{CS}$ المعبر للكتابة أو القراءة من إحد المعابر أو من سجل التحكم، ويوصل عادةً إلى خرج دارة تفكيك العنونة والانتخاب. يملك المعبر أيضاً خطاً للاستهلال، يوصل عادةً إلى خط استهلال المعالج الصغري Reset، بحيث يقوم بإعادة تهيئة المعابر كافة إلى وضع

ابتدائي محدد تكون فيه كافة المعابر معدة كمعابر دخل، وذلك منعاً لإخراج أية قيمة على معبر قد يكون مربوطاً إلى جهاز محيطي يستخدمه كمعبر دخل، لتفادي أي تخريب ممكن للدارات المترابطة.



الشكل 7: البنية الداخلية العامة للمعبر القابل للبرمجة Intel 8255.

2-3 أنماط عمل الدارة 8255A

النمط 0:

يمكن تهيئة المعبر للعمل بالنمط 0 في حال عمليات الدخل/الخرج البسيطة الوحيدة الاتجاه التي لاتحتاج إلى مصافحة. عند تهيئة كلا المعبرين A و B للعمل في النمط 0 يمكن استخدام نصفي المعبر C كمعبر وحيد ثُمانيّ الخطوط، أو كمعبرين رباعيي الخطوط مستقلين.

النمط 1:

يستخدم هذا النمط عند الحاجة إلى استخدام إحد المعبرين A أو B أو كليهما في عمليات دخل/خرج محكومة بالمصافحة. في هذه الحالة تستخدم بعض خطوط المعبر C كخطوط مصافحة. تخصص الخطوط C_0 و C_1 و C_2 للمصافحة الخاصة بالمعبر B إذا هيئت للعمل بالنمط 1، على حين تخصص الخطوط C_3 و C_4 و C_5 للمصافحة الخاصة بالمعبر A إذا هيئت للعمل كمعبر دخل في النمط 1. وعند تأهيل C_3 لتوليد المقاطعة يجب وضع الخانة C_4 على القيمة '1'، كذلك يجب وضع الخانة C_2 على القيمة '1' عند تأهيل C_0 لتوليد المقاطعة. ويبقى الخطان C_6 و C_7 للاستخدام في الإدخال أو الإخراج.

أما إذا هيئ المعبر A للعمل كمعبر خرج في النمط 1 فعندها تعمل الخطوط C_3 و C_6 و C_7 كخطوط المصافحة الخاصة بالمعبر A ويبقى الخطان C_4 و C_5 للاستخدام كخطوط دخل أو خرج. ويجب وضع الخانة C_6 على القيمة '1' لتأهيل الخط C_3 لتوليد المقاطعة.

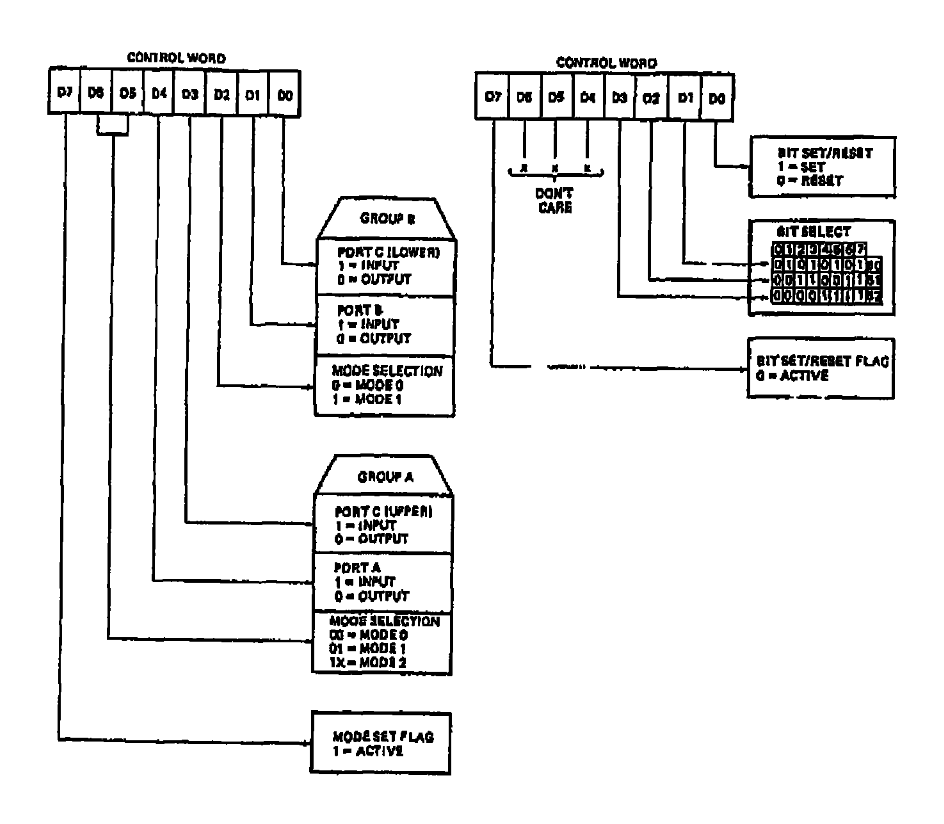
النمط 2:

يمكن تهيئة المعبر A فقط في هذا النمط. يمكن استخدام المعبر A في هذا النمط كمعبر ثنائي الاتجاه محكوماً بالمصافحة، أي أنه يمكن

إدخال أو إخراج المعطيات على نفس خطوط المعبر الثمانية. عند تهيئة المعبر A في هذا النمط، تستخدم الخطوط C_3 و C_4 و C_5 و C_6 و C_7 للمصافحة الخاصة بالمعبر A. ويمكن استخدام الخطوط الثلاثة الباقية من المعبر C للمصافحة الخاصة بالمعبر B إذا هيئت للعمل بالنمط 1.

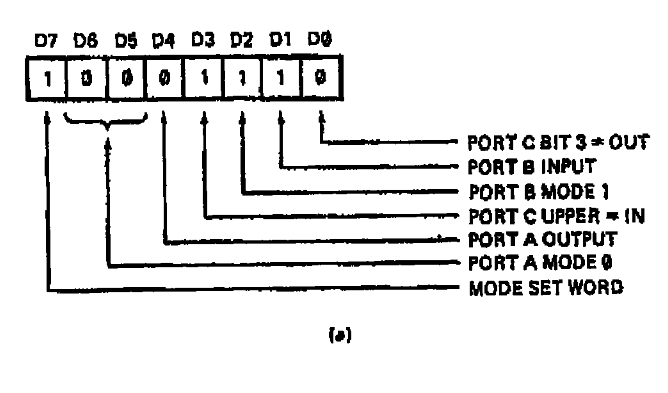
3-3 كلمة التحكم في الدارة 8255A

لاختيار نمط عمل المعابر واتجاهها، ترسل ثمانيّة إلى سجل التحكم وفق الهيئة المبينة في الشكل 8.

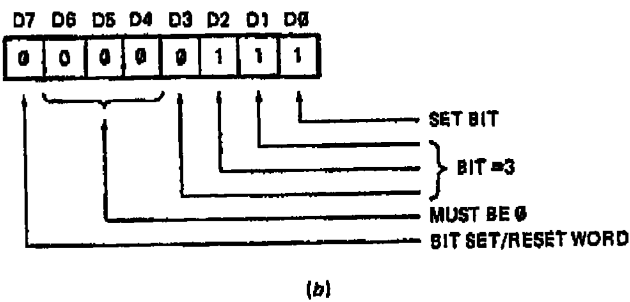


الشكل 8: هيئة كلمة التحكم عند ضبط نمط العمل.

وفي هذه الحالة، تكون الخانة D_7 من كلمة التحكم على القيمة '1' للدلالة على التحكم بالنمط. أما عند استخدام كلمة التحكم للتحكم في القيم الموضوعة على خانات المعبر C، فتوضع الخانة D_7 من كلمة التحكم على القيمة '0' للدلالة على التحكم في خانات المعبر C ويتم التحكم بكل خانة على حدة. يكتب في الخانة D_0 القيمة '0' لوضع الخانة المختارة على القيمة صفر، أو القيمة '1' لوضع الخانة المختارة على القيمة واحد. وتدل القيمة المكتوبة في الخانات D_1 و D_2 و D_3 من كلمة التحكم على الخانة المراد تغيير قيمتها، كما يبين الجدول في الشكل 9.



(a)



(b)

الشكل 9: هيئة كلمة التحكم عند تغيير قيم خانات المعبر C.

أما الخانات الأخرى من كلمة التحكم (D_4 و D_5 و D_6) فتوضع على القيمة 0.

يمكن، عند قراءة المعبر C، معرفة حالة المعبرين A وB إذا كان إحدهما أو كلاهما مبرمجاً للعمل بأحد أنماط المصافحة. فإذا كان المعبر مهيئاً للعمل في النمط 1 وكمعبر دخل، تعطي الخانة C_0 حالة المقاطعة، وتكون قيمتها مساوية للواحد إذا كان هناك طلب للمقاطعة من هذا المعبر وكانت المقاطعة لهذا المعبر مؤهلة (أي $C_2=1$). على حين تعطي الخانة C_2 حالة تأهيل المقاطعة للمعبر B. أما الخانة C_1 فتمثل إشارة الإشعار ACK.

أما إذا كان المعبر B مهيأً للعمل في النمط 1 وكمعبر خرج فتعطي كل من C_0 و C_2 نفس المعلومات، أما C_1 فتمثل إشارة القدح $\overline{STB}$.

كذلك حال المعبر A: فعندما يعمل في النمط 1 كمعبر دخل تعطي الخانة C_3 حالة المقاطعة إذا كانت المقاطعة من المعبر A مؤهلة (أي $C_4=1$)، وتعطي C_4، بطبيعة الحال، حالة تأهيل المقاطعة. أما C_5 فتمثل في هذه الحالة إشارة الإشعار باستقبال المعطيات.

وفي حال كون المعبر A مهيأً للعمل في النمط 1 كمعبر خرج، تبقى إشارة المقاطعة على الخانة C_3 ولكن خانة تأهيل المقاطعة في هذه الحالة هي C_6، وتمثل الخانة C_7 إشارة القدح المترافقة مع وضع المعطيات الجديدة على المعبر.

4 المتحكم المبرمج في المقاطعة

كما سبق وذكرنا في معرض الحديث عن بنية المعالجات 8085 و 8086 وجدنا أن للمعالج مدخلين للمقاطعة «المادية» Hardware Interrupt هما المدخلُ INTR للمقاطعة القابلة للحجب و NMI للمقاطعة غير القابلة للحجب اللذان يسمحان لإشارة خارجية بمقاطعة تنفيذ البرنامج (وهذا ما شرحناه باقتضاب في معرض الحديث عن عمليات الدخل/الخرج). يملك المعالج في داخله قلاباً يسمح بتأهيل أو حجب المقاطعة INTR، فإذا كتبت فيه القيمة 0 تحجب المقاطعة ولا يستجيب المعالج لأية إشارة على الدخل INTR، في حين يكون المعالج مؤهلاً لأن يُقاطع في حال كتابة القيمة '1' في القلاب المسمى براية المقاطعة IF. وهناك تعليمات خاصة بشحن القيمة واحد أو صفر في هذا القلاب، كما سنرى عند الحديث عن البرمجة بلغة المجمّع (الفصل الثالث).

عند الإقلاع تكون قيمة هذا القلاب صفراً، وكذلك يجري إعادة قيمته إلى الصفر آلياً عند استجابة المعالجة لمقاطعة خارجية، لمنع مقاطعة إجرائية تخديم المقاطعة. ولكن إذا كانت المقاطعة المخدّمة ذات أولوية دنيا، يمكن عندها تأهيل المقاطعة في بداية إجرائية تخديم المقاطعة، وذلك للسماح لمقاطعة ذات أولوية أعلى بالاستحواذ على تخديم المعالج. تقوم تعليمة العودة من المقاطعة IRET في نهاية إجرائية تخديم المقاطعة بإعادة تأهيل المقاطعة آلياً، أما المقاطعة NMI فهي مؤهلة دوماً. تسبب جبهة صاعدة على المدخل NMI حدوث المقاطعة، على حين تقدح المقاطعة على المدخل INTR، عندما تكون مؤهلة، عند تحسس مستوى منطقي عالٍ على هذا المدخل. توضع إجرائية خدمة المقاطعة للمقاطعة غير القابلة للحجب في العنوان 0008-000Bh وتملك بعض المعالجات عدداً من مداخل المقاطعة لها أولويات مختلفة وطرق معالجة متباينة. أما في المعالج 8086 فلا يوجد

سوى مدخلي INTR و NMI المذكورين آنفاً. ولكن يستطيع المعالج 8086 أن ينفذ 256 نوعاً من المقاطعات (لكل منها أولوية وإجرائية تخديم المقاطعة الخاصة بها)، ولكي يتمكن المعالج من تمييز أنواع المقاطعة المختلفة نحتاج إلى دارة تسمى بالمتحكم المبرمج في المقاطعة، كالدارة 8259A التي سنأتي على شرحها لاحقاً.

يقوم المعالج 8086، استجابة لطلب المقاطعة، بـ «الانفكاك» عن المسرى[2]؛ ثم يرسل نبضة إشعار باستلام المقاطعة على الخط $\overline{INTA}$ (ليُعلم الجهاز الخارجي باستلام المقاطعة). يعود المعالج بعدها ليرسل نبضة ثانية على نفس خط إشعار المقاطعة ليطلب من الجهاز الخارجي أن يضع رقماً يعبر عن نوع المقاطعة المطلوبة (من 0 إلى 255) على خطوط المعطيات الثمانية الأدنى لكي يقرأها المعالج.

فور قراءة المعالج لنوع المقاطعة، يُستخدم هذا الرقم كموجه إلى إجرائية تخديم المقاطعة، وبذلك يمكن استخدام مدخل المقاطعة لتخديم أكثر من جهاز خارجي، لكل منها نوع من أنواع المقاطعة وعنوان إجرائية تخديم المقاطعة خاص به.

1-4 المقاطعات المتعددة

ولكن ماذا لو ورد إلى المعالج أكثر من طلب للمقاطعة؟ في هذه الحالة سوف يقوم المعالج بتخديم المقاطعة ذات الأولوية العليا أولاً. وتملك المقاطعات البرمجية الأولوية العليا وتليها في الأولوية المقاطعة غير القابلة للحجب NMI وتليها في الأولوية المقاطعة الناجمة عن الخط INTR.

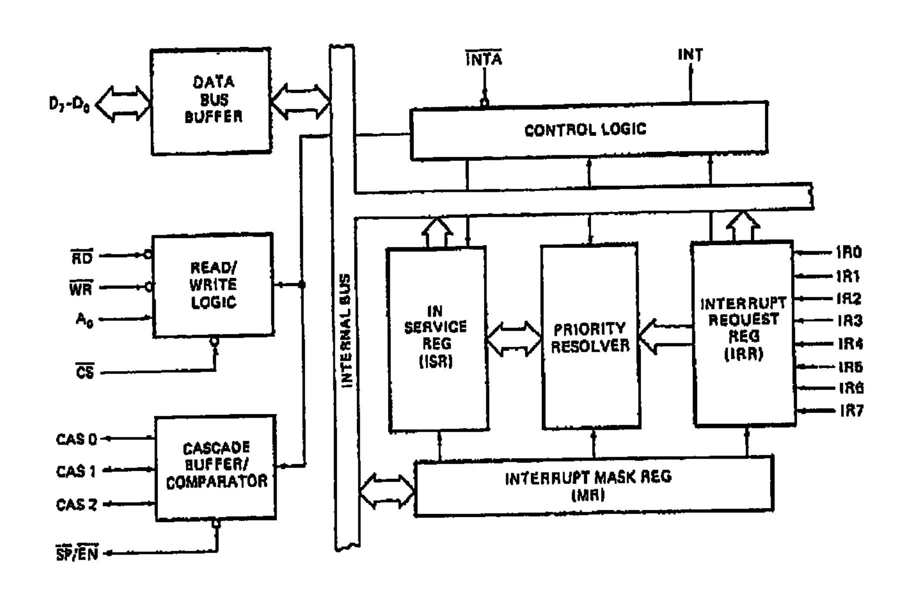
لنذكّر، قبل استعراض المُتحكم في المقاطعة، بالعمليات التي يقوم بها المعالج 8086 إذا استقبل إشارة ذات منطق عال على المدخل INTR وكانت المقاطعة مؤهلة (أي راية المقاطعة IF تحتوي على القيمة '1'):

2 نعبر عن ذلك بالقول إن المعالج يضع خطوط المعطيات في حالة الممانعة العالية.

1 دفع المؤشرات Flags إلى المكدس.
2 وضع القيمة '0' في قلاب تأهيل المقاطعة وراية تأهيل التنفيذ الخطوي TF.
3 دفع عنوان العودة إلى المكدس.
4 وضع مسرى المعطيات في نمط الدخل.
5 إرسال نبضتي إشعار باستلام المقاطعة على المربط $\overline{INTA}$ مهمتهما الطلب من الجهاز الخارجي (كالمتحكم 8259A) إعلام المعالج برقم المقاطعة الحاصلة.
6 عندما يستلم المعالج رقم المقاطعة يقوم بضربه بأربعة ليولد عنواناً يشير إلى موقع في جدول عناوين المقاطعة.
7 يأخذ المعالج من الموقع المعنون في جدول متجهات (عناوين) المقاطعة والمواقع الثلاثة اللاحقة، العنوان الدال على عنوان المقطع CS ومقدار الانزياح IP اللازمين للحصول على عنوان إجرائية خدمة المقاطعة.

يبين الشكل 10 المخطط الصندوقي للدارة 8259A. تسمح خطوط مسرى المعطيات للمعالج بإرسال كلمات التحكم اللازمة لبرمجة الدارة وبقراءة سجل الحالة من الدارة 8259A. تجري هذه العملية باستخدام خطوط التحكم $\overline{RD}$ و $\overline{WR}$، وذلك عندما يكون خط انتخاب الدارة 8259A على المنطق المنخفض. كذلك تنقل الدارة 8259A رقم المقاطعة إلى المعالج عبر مسرى المعطيات. للدارة 8259A ثمانية مداخل مختلفة للمقاطعة IR0 وحتى IR7؛ وفي حال كانت الدارة مؤهلة وأتتها إشارة مقاطعة على أي من المداخل السابقة متمثلة بمنطق عال فإنها سوف ترسل إشارة عالية على المخرج INTR. فإذا كانت الدارة مؤهلة وأتتها إشارة مقاطعة على أيٍ من المداخل السابقة، متمثلة بمنطق عال، فإنها سوف ترسل إشارة عالية إلى المخرج INTR. وإذا كانت هذه الإشارة موصولة إلى مدخل المقاطعة للمعالج وكانت المقاطعة مؤهلة، فسوف تسبب هذه الإشارة مقاطعة المعالج وتنفيذ الاستجابة الموصوفة سالفاً.

يوصل المدخل $\overline{INTA}$ للدارة 8259A إلى مخرج $\overline{INTA}$ للمعالج، وتستخدم الدارة النبضة الأولى للقيام ببعض العمليات التي تعتمد على النمط الذي برمجت الدارة وفقه، على حين تؤدي نبضة الإشعار الثانية إلى قيام الدارة بإخراج رقم المقاطعة على خطوط مسرى المعطيات الثمانية. يعتمد رقم المقاطعة المرسل على رقم المدخل الذي ولّد المقاطعة IR وعلى معامل آخر يعتمد على البرمجة والإعداد الأوليين للدارة.



الشكل 10: المخطط الصندوقي للدارة 8259A.

يمكن أن تعمل الدراة 8259A بعدة أنماط يمكن التحكم فيها برمجياً عن طريق كلمة التحكم التي يرسلها المعالج إلى الدارة. ويعد نمط الأولوية الثابتة هو نمط الاستخدام الأكثر شيوعاً، وتكون فيه الأولوية العليا للمقاطعة الآتية من المدخل IR0 وتليها تلك الآتية من IR1 وهكذا حتى IR7، التي تملك أدنى أولوية. يمكن حجب أية مقاطعة عن طريق سجل حجب المقاطعة بإرسال كلمة تحكم لهذا السجل تحوي صفراً في الخانة المقابلة لمدخل المقاطعة المرغوب في تأهيله. يكتب في

سجل طلب المقاطعة IRR القيمة '1' في الخانة الموافقة للمداخل التي أتى منها طلب المقاطعة إن لم تكن هذه المقاطعة محجوبة، على حين يشير سجل خدمة المقاطعة ISR إلى المقاطعات التي يقوم المعالج حالياً بتخديمها. أما مهمة وحدة حل الأولوية فهي الإيعاز للبدء بتخديم المقاطعات وفق الأولوية المعتمدة.

لنفترض أن المقاطعتين IR4 و IR2 كانتا مؤهلتين، ووردت إشارة مقاطعة إلى المدخل IR4، عندها تصبح قيمة الخانة 4 من سجل طلب المقاطعة '1' وتكتشف وحدة حل الأولوية ذلك. ولكي تتمكن هذه الوحدة من معرفة الإجراء الواجب اتخاذه، تفحص هذه الوحدة سجل تخديم المقاطعة ISR لترى أهناك مقاطعات ذات أولوية أعلى يجري تخديمها أم لا. في حال وجود مقاطعة ذات أولوية أعلى يجري تخديمها فعلاً (ويشير إلى ذلك القيمة '1' للخانة المقابلة للمقاطعة في سجل تخديم المقاطعة)، عندها لا تقوم وحدة حل الأولوية بأي إجراء. أما إذا لم توجد أية مقاطعة ذات أولوية أعلى يجرى تخديمها، فتقوم وحدة حل الأولوية بإرسال إشارة المقاطعة للمعالج. وعندما يستجيب المعالج بإرسال نبضات إشعار باستلام المقاطعة، ترسل الدارة 8259A رقم المقاطعة الموافق للمقاطعة IR4 والمحدد عند تهيئة الدارة. يستخدم المعالج هذا الرقم ليجد عنوان الإجرائية المكتوبة لتخديم المقاطعة IR4 لتنفيذها.

لنفترض ورود إشارة مقاطعة ثانية IR2 أثناء تنفيذ المعالج لإجرائية تخديم المقاطعة IR4. لما كانت IR2 مؤهلة (غير محجوبة) تصبح قيمة الخانة 2 من سجل طلب المقاطعة '1' وتكتشف وحدة حل الأولوية ذلك، وتتفحص سجل تخديم المقاطعة لتقرير ما يجب عمله. فإذا وجدت خانة مساوية للواحد مقابلة لمقاطعة ذات أولوية أعلى،

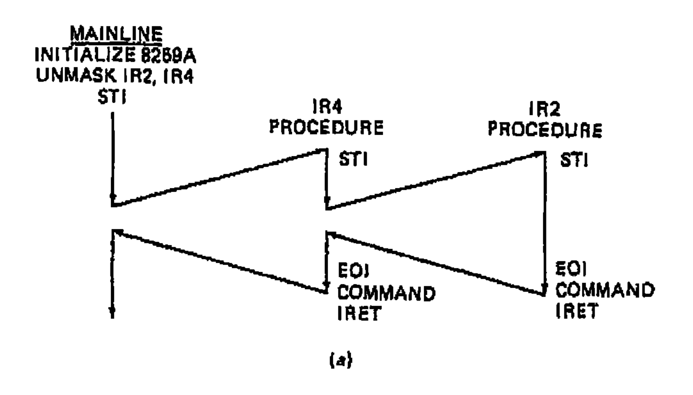
عندها يكون المعالج مشغولاً بتخديم مقاطعة ذات أولوية أعلى، فينتظر المعالج انتهاء هذا التخديم ووضع الخانة المقابلة في سجل تخديم المقاطعة على القيمة '0'. أما إذا وجدت وحدة حل الأولوية أن هذه المقاطعة هي ذات الأولوية العليا، فتضع الخانة المقابلة لهذه

المقاطعة في سجل تخديم المقاطعة على القيمة '1'، وترسل إشارة المقاطعة للمعالج، كما هو الحال في هذا المثال. إذا أعيد تأهيل المقاطعة في بداية إجرائية تخديم المقاطعة IR4، كما هو مبين بالشكل 11، فسوف تقاطع إشارة INT المعالج ثانية. وعندما يرسل المعالج نبضات إشعار باستلام المقاطعة، ترسل الدارة 8259A الرقم الخاص بالمقاطعة IR2، الذي يستخدم لتحديد عنوان إجرائية تخديم IR2. وفي نهاية إجرائية تخديم المقاطعة IR2 يرسل المعالج كلمة تحكم للدارة 8259A لوضع القيمة '0' في الخانة المقابلة للمقاطعة IR2 في سجل تخديم المقاطعة، بحيث يمكن استئناف تخديم المقاطعات ذات الأولوية الدنيا، ثُم تعيد تعليمة العودة من المقاطعة IRET التنفيذ إلى إجرائية تخديم المقاطعة IR4 حيث توقفت؛ وبدورها تعيد تعليمة IRET في نهاية إجرائية تخديم المقاطعة IR4 التنفيذ إلى البرنامج الأساسي حيث حدثت مقاطعته. أما إذا لم نقم بإعادة تأهيل المقاطعة في بداية إجرائية تخديم المقاطعة (بتعليمة STI) فلن يستجيب المعالج لإشارة المقاطعة الصادرة عن طلب المقاطعة IR2 حتى تنتهي إجرائية تخديم المقاطعة السابقة IR4.

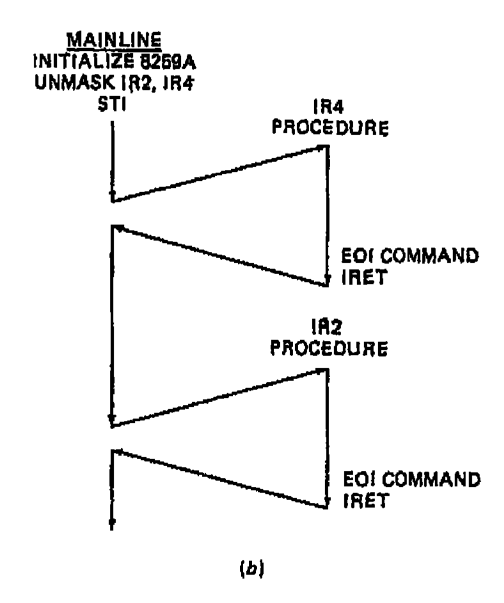
2-4 ربط الدارة 8259A إلى النظام

يبين الشكل 12 ربط الدارة 8259A#1 إلى بطاقة تطوير من النوع SDK-86. يقوم مفكك الترميز 74LS138 بانتخاب الدارة 8259A عندما يوضع العنوان FF00h على مسرى عناوين المعالج. ويستخدم الدخل A0 من الدارة 8259A#1، الموصول إلى الخط A1 من خطوط عنونة المعالج، لانتخاب أحدالعنوانين الداخليين؛ وبذا يصبح عنوان هذين السجلين هو FF00h و FF02h. وترتبط خطوط المعطيات الثمانية من الدارة 8259A بالخطوط الثمانية الدنيا من مسرى المعطيات للمعالج 8086، حيث يتوقع المعالج استلام رقم المقاطعة على هذه الخطوط. وترتبط خطوط القراءة والكتابة للدارة بخطوط القراءة والكتابة

للمعالج $\overline{WR}$ ، $\overline{RD}$ وكذلك يربط خط إشعار المقاطعة للمعالج بمخرج إشعار المقاطعة للدارة $\overline{INTA}$ على حين يربط خط المقاطعةللمعالج INTR بخط طلب المقاطعة للدارة INT. أما المدخل $\overline{SP}/\overline{EN}$ فيربط إلى المنطق العالي بسبب استخدام دارة 8259A وحيدة، وكذلك لاتربط النقاط CAS0, CAS1, CAS2 بأي نقطة أخرى.

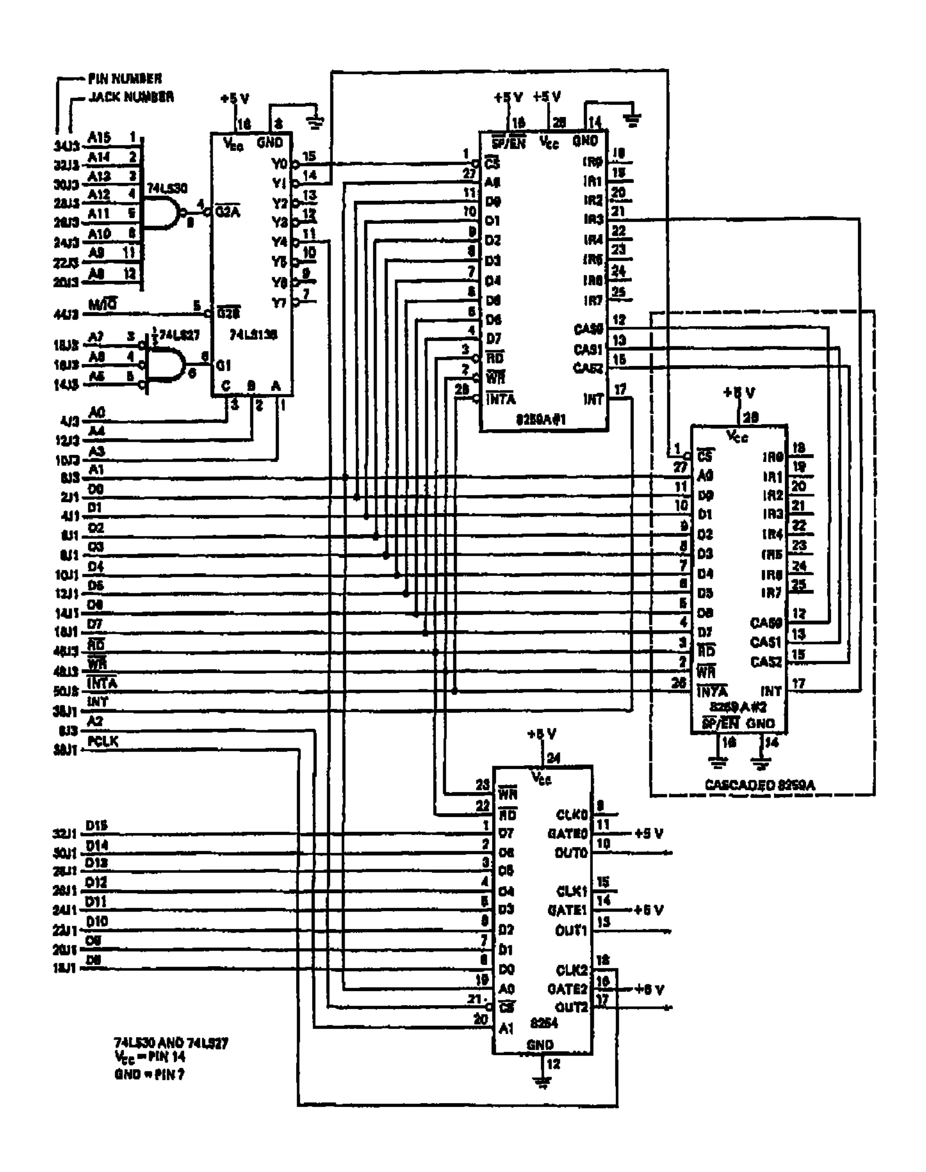


(a)



(b)

الشكل 11: معالجة ورود مقاطعتين متتاليتين.



الشكل 12: مثال عن نظام معالج صغري مع وحداته المحيطية – البطاقة SDK-86.

في هذه الحالة يتيح لنا النظام ثمانية خطوط للمقاطعة. ويجب ربط كافة مداخل المقاطعة غير المستخدمة بالجهد الأرضي لمنع حدوث مقاطعة بسبب إشارات الضجيج التي قد تتراكب على الخطوط. يمكن إضافة دارة متحكم في المقاطعة ثانية من النوع 8259A في المساحة المنقطة والمشار إليها بـ 8259A#2، وهذا ما يوسع عدد مداخل المقاطعة إلى 16 خطاً. في هذه الحالة يجب إسناد عنوان إلى الدارة الثانية، مختلفٍ عن عنوان الدارة الأولى.

ولما كان المعالج 8086 يملك خط مقاطعة وحيد، وجب أن ترتبط به مباشرة دارة 8259A وحيدة تعمل كدارة تحكم في المقاطعة حاكمة Master، على حين يربط خرج المقاطعة INT للدارة التابعة Slave بأحد مداخل طلب المقاطعة للدارة الحاكمة، وبذا يمكن وصل ثمان دارات تابعة إلى الدارة الحاكمة 8259A#1، وبالتالي يمكننا الحصول على 64 خط مقاطعة، على حين يتصل خط إشعار المقاطعة بكافة دارات التحكم في المقاطعة المتصلة (سواء أكانت حاكمة أم تابعة). كذلك يسند إلى كل دارة تابعة عنوانان خاصان بسجل التحكم والحالة للدارة. يربط المدخل $\overline{SP}/\overline{EN}$ للدارات التابعة إلى الأرضي لكي تعرف الدارة أنها تعمل في نمط التابع، أما خطوط القراءة والكتابة والمعطيات فتربط إلى خطوط القراءة والكتابة والمعطيات الخاصة بالمعالج على الترتيب، وتربط خطوط الربط المتتالي CA2, CAS1, CAS0 من الدارة الحاكمة إلى الخطوط المقابلة من الدارة التابعة. تعمل مرابط الربط المتتالي كمخارج في الدارة الحاكمة، وتعمل كمداخل في الدارات التابعة.

عند استلام الدارة التابعة لإشارة مقاطعة على أحد مداخلها، وإذا كانت هذه المقاطعة غير محجوبة وذات أولوية أعلى من المقاطعة المخدمة للدارة التابعة نفسها، عندها ترسل الدارة التابعة إشارة مقاطعة INT إلى أحد مداخل المقاطعة للدارة الحاكمة. إذا لم تكن هذه المقاطعة محجوبة وذات أولوية أعلى من المقاطعة التي يجري تخديمها في الدارة الحاكمة، ترسِل الدارة الحاكمة إشارة INT على خط المقاطعة

INTR للمعالج. وإذا كانت المقاطعة مؤهلة في المعالج، يدخل المعالج في دورة تخديم للمقاطعة، ويرسل نبضتي إشعار باستلام المقاطعة لكلتا الدارتين الحاكمة والتابعة. تتجاهل الدارة التابعة أول نبضة إشعار، على حين تقوم الدارة الحاكمة عند استلامها للنبضة الأولى بإخراج عنوان مكون من ثلاث خانات، للدلالة على الدارة التابعة المولدة للمقاطعة على خطوط الربط المتتالية CA2, CAS1, CAS0 (يسند عنوان فريد إلى كل دارة تابعة عند التهيئة الأولية لهذه الدارات). يؤهل هذا العنوان الدارة التابعة المعنية بحيث تستقبل نبضة إشعار المقاطعة الثانية من المعالج، وترسل رقم المقاطعة المطلوبة إلى المعالج على خطوط المعطيات الثمانية.

يمكن العودة إلى نشرة معلومات الدارة 8259A لمعرفة تفاصيل التهيئة وشكل كلمات التحكم الواجب إرسالها لبرمجتها بنمط العمل المطلوب.

5 المؤقت/العداد المبرمج

من أبسط أمثلة استخدام المقاطعة الاستفادة من مدخل المقاطعة في العد وقياس الزمن والتوقيت. بالطبع يستطيع المستثمر كتابة برنامج يحوي حلقة تأخير لتحقيق التواقت في بعض التطبيقات، إلا أن هذا يعني أن المعالج الصغري لايقوم بأي عمل مفيد ما دام في حلقة التأخير البرمجية. لذا فمن المفيد أكثر، وصل دارة مؤقت خارجية إلى أحد مداخل المقاطعة للمعالج الصغري، مثلاً إذا وضعنا المعالج الصغري في منظومة لقياس درجة الحموضة لسائل ما، بحيث يجب قراءة درجة الحموضة من إحد معابر الدخل/الخرج كل أربع دقائق؛ في هذه الحالة يمكن وضع دارة توقيت بسيطة مكونة من مهتز يولد موجة مربعة مطالها 0-5V، ودورها أربع دقائق توصل إلى مدخل المقاطعة NMI على سبيل المثال.

يمكن أيضاً أن يقوم المعالج بقياس الوقت الفعلي (ساعة الزمن الحقيقي Real-Time Clock) إذا ربطنا مهتزاً دقيقاً بعمل بتردد 1Hz، إلى ما هنالك من التطبيقات... ولتوليد إشارة ساعة بتردد دقيق (كذاك المطلوب لساعة الزمن الحقيقي) تستغل عادة دارة المهتز الذي يولد نبضات الساعة اللازمة لعمل المعالج الصغري، لأنها عالية الدقة والاستقرار؛ ولكن ترددها عالٍ جداً ولا يصلح لمعظم تطبيقات التوقيت، كما أنه، ولنفس السبب، لايمكن ربطه إلى مدخل المقاطعة مباشرة. لذا نلجأ إلى تقسيم تردد ساعة المعالج باستخدام عناصر خارجية للحصول على التردد المناسب للتوقيت/المقاطعة.

تحوي معظم نظم المعالجات الصغرية عدادات مثل 8253 أو 8254 من شركة INTEL قابلة للبرمجة من قِبل المعالج، لكي تقوم بتقسيم ساعة المعالج على أي رقم بحيث نحصل على التردد المطلوب. ويمكن استخدام هذه العناصر في تطبيقات هامة أخرى (كالعد)، إضافةً إلى عملها كمقسمات تردد قابلة للبرمجة.

سنستعرض فيما يلي البنية العامة للمؤقتات/العدادات 8253 و 8254 وأنماطَ عملها وطريقة تهيئتها وبرمجتها بوجهٍ عام. ويمكن العودة لاحقاً إلى نشرة معلومات الدارة من أجل تفاصيلِ البرمجة وهيئة كلمات التحكم الواجب إرسالها إلى الدارة لتشغيلها في نمط معين يناسب التطبيق المعني.

1-5 بنية الدارات 8253 و 8254

تحوي الدارات 8253 و 8254 ثلاثة عدادات كل منها بـ 16 خانة اثنانية يمكن برمجتها للعمل في عدة أنماط مختلفة. والدارتان 8253 و 8254 متوافقتان من حيث المرابط ومتطابقان من حيث الوظيفة.

يبين الشكل 13 المخطط الصندوقي الموضح لبنية الدارة 8254. وهي تحوي ثلاثة عدادات شبيهة بالعدادات المنطقية التي دُرست في الجزء الأول من هذا الكتاب، إلا أنها تمتاز عنها بإمكان شحن قيمة ما

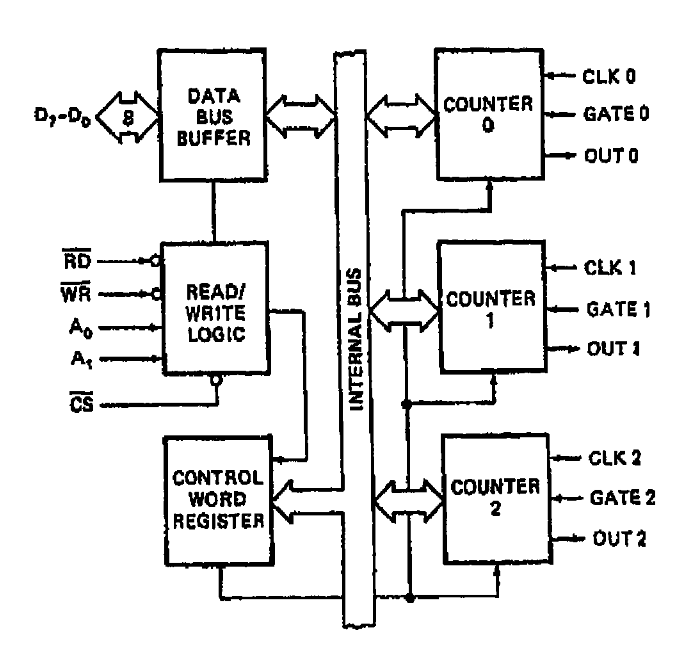
في العدادات، وإطلاق العد أو إيقافه بتعليمات برمجية، لذا فهي قابلة للبرمجة؛ ويقوم البرنامج بإرسال الكلمات الحاوية لقيمة العد الابتدائي وكلمات التحكم، إلى العداد المبرمج كما لوكان يكتب كلمات المعطيات تلك إلى معبر دخل/خرج. تظهر في الجانب الأيسر من الشكل خطوط الإشارات التي تُستخدم لربط العداد إلى مسرى النظام، وهي خطوط مسرى المعطيات الثمان وإشارة انتخاب $\overline{CS}$ تربط عادةً إلى مفكك ترميز العنونة، إضافةً إلى خطي العنونة A0 و A1 اللذين يسمحان بعنونة أحد العدادات الثلاثة أو سجل كلمة التحكم الموجودة ضمن الدارة. على حين يظهر في الجانب الأيمن من الشكل مداخل ومخارج العدادات، ويمكن تطبيق إشارات ترددها من 0 حتى 8 MHz في حالة الدارة 8254 (أو حتى 2.6 MHz في حالة الدارة 8253) على مداخل العد للعدادات المشار إليها بـ CLK على المخطط. يسمح المدخل المسمى GATE بحجب العد أو تأهيله بواسطة إشارة خارجية، فعندما تكون الإشارة على المدخل GATE ذات مستوى منطقي عال يكون العداد مؤهلاً للعد، ويكون العد محجوباً إذا كانت هذه الإشارة ذات مستوى منطقي منخفض. يرمز إلى مخارج العدادات بـ OUT.

5-2 ربط المؤقت/العداد المبرمج 8254 إلى النظام

يبين الشكل 12 كيفية ربط دارة 8254 إلى نظام معالج صغري وكيفية إضافة متحكم في المقاطعة إلى النظام، كما شرحنا في فقرة المتحكم في المقاطعة المبرمج. يستخدم مفكك الترميز 74LS138 لتوليد إشارة الانتخاب $\overline{CS}$ للدارة 8254 ودارات أخرى.

يتضح من الشكل المذكور آنفاً أن الناخب 74LS138 يقوم بانتخاب دارة المؤقت/العداد 8254 عند أية عملية إدخال/إخراج من العنوان القاعدي FF01h (عندما يكون الخط $M/\overline{IO}$ على المنطق المنخفض). ولما كان خطا العنونة للمعالج A1 و A2 متصلين بخطي العنونة الخاصين

بالدارة 8254، فإن سجلات الدارة تأخذ المواقع FF07h, FF05h, FF03h, FF01h وتقابل سجلات العداد 0 والعداد 1 والعداد 2 وسجل التحكم. نلاحظ أن العناوين فردية وقد وُصلت خطوط المعطيات الثمان العليا من المعالج إلى خطوط معطيات المؤقت/العداد المبرمج. تتصل كذلك خطوط القراءة والكتابة $\overline{RD}$ و $\overline{WR}$ للدارة بخطوط القراءة والكتابة للنظام الصادرة عن المعالج.



الشكل 13: البنية الداخلية للمؤقت/العداد 8254.

3-5 تهيئة المؤقت/العداد المبرمج 8254

تأخذ قلابات العدادات والسجلات الداخلية فيها عادة قيماً عشوائية عند تطبيق التغذية عليها لأول مرة. لذا تجب تهيئتها ووضعها في نمط العمل المرغوب فيه للتطبيق المعني. وكما سبق

ورأينا عند استعراض تهيئة متحكم المقاطعات المبرمج، يجب، للقيام بعملية التهيئة على وجهٍ صحيح وبلا هفوات، تنفيذ خطوات معينة:

1 تحديد العنوان القاعدي للجهاز أو الدارة، انطلاقاً من طريقة وصل خطوط العنونة من المعالج إلى مفكك الترميز، وخرْج مفكك الترميز إلى الجهاز أو الدارة. (وجدنا أن العنوان القاعدي للمؤقت/العداد المبرمج في مثالنا الوارد في الشكل 12 هو FF01h).

2 استخدام نشرة المعلومات للدارة المعنية لتحديد العناوين الداخلية لكل من سجلات التحكم أو المعابر أو المؤقتات أو سجلات الحالة، الخ... في الدارة 8254 يقابل العنوان A1A0=00 العداد Counter 0، ويقابل العنوان A1A0=01 العداد Counter1، ويقابل العنوان A1A0=10 العداد Counter 2، ويقابل العنوان A1A0=11 سجل كلمة التحكم.

3 إضافة العنوان الداخلي إلى العنوان القاعدي لتحديد عنوان كل سجل من سجلات الجهاز بالنسبة إلى النظام. ومن ثم يصبح للعدادات الثلاثة ولسجل التحكم في مثالنا العناوين: FF07h, FF05h, FF03h, FF01h وفق التوصيل المبين بالشكل 12.

4 العودة إلى نشرة معلومات الجهاز أو الدارة لتحديد شكل كلمات التحكم الواجب إرسالها إلى الجهاز لتهيئته للعمل بالطريقة المطلوبة. يبين الشكل 14 هيئة كلمات التحكم أو الأوامر الواجب إرسالها إلى سجل التحكم، لإعداد كل عداد من العدادات الموجودة ضمن الدارة 8254 في النمط المطلوب.

5 تشكيل كلمات التحكم الواجب إرسالها وفق الهيئة الواردة في النشرة وتبعاً لنمط التشغيل المطلوب. يمكن كتابة معنى كل خانة من كلمة التحكم لتسهيل التدقيق لاحقاً.

6 أخيراً إرسال كلمة التحكم، إلى سجل التحكم وإرسال القيمة الابتدائية إلى العداد المعني. يجب إرسال كلمة تحكم من أجل كل عداد من العدادات الموجودة في الدارة 8254، وبما أنه لا يوجد في الدارة 8254 إلا سجل تحكم وحيد، ترسل كلمة التحكم الخاصة بكل

عداد من العدادات إلى نفس العنوان، ولكن نستخدم أعلى خانتين من كلمة التحكم SC1 SC0 لتحديد العداد الذي نرغب في إعداده بكلمة التحكم تلك. تمكننا الخانة الدنيا، المسماة BCD، من برمجة العداد للعد تنازلياً من القيمة المشحونة فيه إما اثنانياً على 16 خانة اثنانية أو بالترميز العشري المرمز اثنانياً وعلى أربع خانات عشرية. عند وضع 0 في الخانة من كلمة التحكم يعامل العداد المعني القيمة التي تشحن فيه كعدد اثناني، أما إذا وضعت القيمة 1 في الخانة D0 فيعامل العداد هذه القيمة على أنها عدد عشري مرمز اثنانياً (يقع بين 0000 و 9999).

تحدد القيم المكتوبة في الخانات M2, M1, M0 من كلمة التحكم نمط عمل العداد، أي أثر المدخل GATE على العد والإشارة الناتجة على مخرج العداد OUT. وسنستعرض أنماط العمل لاحقاً.

تحدد الخانتان RW0 و RW1 الطريقة التي نرغب في استخدامها لقراءة قيمة العدادات، أو لشحن قيمة ما في هذه العدادات. فعند كتابة القيمة 01 فيهما ضمن كلمة التحكم، فهذا يعني أننا سنقوم بشحن العداد المعني بالكلمة الثُمانيّة الدنيا فقط بإرسالها إلى عنوان العداد المعني. أما إذا كتبت القيمة 10 في خانتي القراءة والكتابة RW1 RW0 فهذا يعني أننا سنشحن الكلمة الثُمانيّة العليا فقط من قيمة العداد بإرسالها إلى عنوان العداد المعني. وأما إذا كتبت القيمة 11 في خانتي القراءة والكتابة فهذا للدلالة على أننا سنشحن العداد المعني بقيمة ذات 16 خانة اثنانية على مرحلتين، بإرسال الثُمانيّة الدنيا من قيمة العد أولاً إلى عنوان العداد، ثم بإرسال الثُمانيّة العليا إلى نفس العنوان.

يمكن قراءة العد من أي من العدادات في أي لحظة بإرسال كلمة تحكم إلى سجل التحكم، تكتب في خانتي القراءة والكتابة من كلمة التحكم القيمة RW1 RW0 = 00 فتحفظ قيمة العد في تلك اللحظة في سجل مَسْك داخلي. بعدها يجب إرسال كلمة تحكم ثانية تحدد فيها قيمة الخانتين RW1 RW0 الطريقة التي نرغب وفقها قراءة قيمة العداد المختزنة في سجل اللاقف Latch (الثُمانيّة العليا فقط أو

الثُمانيّة الدنيا فقط، أو القيمة الست عشرية على مرحلتين: الثمانية الدنيا أولاً ثم الثمانية العليا).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC – SELECT COUNTER:

SC1	SC0	
0	0	SELECT COUNTER 0
0	1	SELECT COUNTER 1
1	0	SELECT COUNTER 2
1	1	READ-BACK COMMAND (SEE READ OPERATIONS)

RW – READ/WRITE:

RW1	RW0	
0	0	COUNTER LATCH COMMAND (SEE READ OPERATIONS)
0	1	READ/WRITE LEAST SIGNIFICANT BYTE ONLY.
1	0	READ/WRITE MOST SIGNIFICANT BYTE ONLY.
1	1	READ/WRITE LEAST SIGNIFICANT BYTE FIRST, THEN MOST SIGNIFICANT BYTE.

M – MODE:

M2	M1	M0	
0	0	0	MODE 0 – INTERRUPT ON TERMINAL COUNT
0	0	1	MODE 1 – HARDWARE ONE-SHOT
X	1	0	MODE 2 – PULSE GENERATOR
X	1	1	MODE 3 – SQUARE WAVE GENERATOR
1	0	0	MODE 4 – SOFTWARE TRIGGERED STROBE
1	0	1	MODE 5 – HARDWARE TRIGGERED STROBE

BCD:

0	BINARY COUNTER 16-BITS
1	BINARY CODED DECIMAL (BCD) COUNTER (4 DECADES)

NOTE: DON'T CARE BITS (X) SHOULD BE 0 TO INSURE COMPATIBILITY WITH FUTURE INTEL PRODUCTS.

الشكل 14: هيئة كلمات التحكم اللازمة لبرمجة العدادات.

4-5 أنماط عمل المؤقت/العداد 8254 وتطبيقاته

يمكن برمجة العداد 8254 للعمل في واحد من أنماط العمل الستة المتاحة، التي نستعرضها هنا باختصار. ولمعرفة تفاصيل أكثر وتعرف أشكال إشارات خرج ودخل العدادات في كل من هذه الأنماط، يمكن الرجوع إلى نشرات المعلومات التفصيلية.

النمط 0: المقاطعة عند العد النهائي Interrupt on Terminal Count

في هذا النمط يُنتخب نمط العمل بوضع القيمة 000 في الخانات المقابلة لـ M2 M1 M0 من كلمة التحكم، ثُم يشحن العداد الموافق بالقيمة المراد عدها. يقوم العداد بعد ذلك بعدٍّ النبضات الواردة إلى مدخل الساعة CLK الخاص به تنازلياً ما بقيت إشارة تأهيل العد GATE على الواحد (يتوقف عن العد عندما تكون هذه الإشارة صفراً)، ويتحول خرج العداد OUT من الصفر إلى الواحد فور انتقال قيمة العد إلى 0000h ويبقى على ذلك، في حين يتابع العداد العد التنازلي من 0000h إلى FFFFh، إلا إذا شُحن العداد بقيمة جديدة. يمكن الاستفادة من خرج العداد OUT لمقاطعة المعالج،إذا وُصِل هذا الخرج إلى مدخل مقاطعة يقدح على المستوى العالي أو الجبهة الصاعدة، عند انتقال قيمة العداد إلى الصفر (العد النهائي).

يستخدم هذا النمط لعد عدد معين من الأحداث، ومن ثَم إتخاذ إجراء محدد عند الوصول إلى هذا العدد بواسطة إجرائية المقاطعة، بدلاً من وصل النبضات مباشرة إلى مدخل المقاطعة وإسناد مهمة العد إلى إجرائية تخديم المقاطعة.

النمط 1: توليد نبضة وحيدة قابلة لإعادة القدح Retriggerable One-Shot

للعمل في هذا النمط تكتب القيمة 001 في الخانات المقابلة لـ M2 M1 M0 من كلمة التحكم، ثُم يشحن العداد بالقيمة المراد عدها،

التي تقابل عدد نبضات الساعة التي سيبقى عليها خرج العداد في المستوى العالي. عند كتابة كلمة التحكم في سجل التحكم يصبح خرج العداد الموافق OUT واحداً. في هذا النمط يتصرف المدخل GATE وكأنه مدخل قدح، إذ لا يبدأ العداد بالعد حتى ينتقل المدخل GATE من الصفر إلى الواحد، عندها تنتقل قيمة العد من سجل العداد إلى العداد نفسه، ويبدأ العد بدءاً من نبضة الساعة التالية على مدخل الساعة CLK ويعود الخرج OUT إلى الصفر. ثُم تقوم كل نبضة ساعة تالية بإنقاص قيمة العد المشحونة بمقدار واحد حتى يصل العد إلى القيمة 0000h، فينتقل خرج العداد إلى واحد ثانية. ومن ثَم عند قدح العداد بجعل المدخل GATE عالياً ينتقل خرج العداد إلى المستوى المنطقي المنخفض، ويبقى كذلك مدةً تساوي العدد المشحون في العداد N مضروباً بدور الساعة، أي إن عرض نبضة الخرج على المستوى المنخفض يساوي إلى N دوراً من أدوار الساعة.

إذا وردت نبضة موجبة إلى الدخل GATE قبل انتهاء العد، تعود قيمة العد الأساسي لتُشحن في العداد، ويعود العد التنازلي من هذه القيمة، ويبقى الخرج في هذه الأثناء على المستوى المنطقي المنخفض حتى انتهاء العد (الوصول إلى القيمة 0000h) أو حتى ورود نبضة إعادة قدح ثانية.

أما في حال شحن سجل العداد بقيمة جديدة أثناء العد (أي عندما يكون الخرج OUT على الصفر) فإن القيمة الجديدة لن تنتقل إلى العداد ولن يبدأ العد التنازلي منها حتى ورود نبضة قدح جديدة إلى مدخل القدح GATE.

النمط 2: مولّد إشارات مقاطعة ميقاتية Timed Interrupt Generator

ذكرنا في معرض حديثنا عن المقاطعة تطبيق توليد ساعة زمن حقيقي باستخدام المقاطعة؛ في هذا النمط من العمل يستطيع العداد 8254 أن يقوم بتوليد نبضات ساعة زمن حقيقي بمعدل أخفض بكثير من تردد ساعة عمل المعالج، وأعلى بكثير من 1Hz؛ على هذا يمكن

قياس أزمنة أقصر بكثير من الثانية (أي بتمييزية أعلى) وبدقة عالية. تستخدم معظم نظم الحواسيب ساعة زمن حقيقي بتردد 1 KHz، أي إن الزمن الفاصل بين إشارات المقاطعة الناجمة هو 1 ms.

للعمل في هذا النمط تكتب القيمة 010 في الخانات M2 M1 M0 من كلمة التحكم ثُم يُشحن العداد المراد تشغيله في هذا النمط بالقيمة المطلوبة. وإذا كانت إشارة التأهيل GATE على المستوى العالي، يبدأ العداد بالعد تنازلياً بدءاً من نبضة الساعة التالية لشحن العداد. وعندما تصل قيمة العداد إلى القيمة واحد، ينزل خرج العداد OUT إلى الصفر مدةً تعادل نبضة ساعة واحدة، إذ تتسبب الجبهة الهابطة لنبضة الساعة التالية بانتقال قيمة العداد إلى الصفر وإعادة شحنة بنفس قيمة العد الابتدائية. ويعود العداد للعد تنازلياً وتتكرر هذه الدورة. فإذا شحنا العداد بقيمة N ينتقل خرج العداد OUT إلى الصفر مدةَ دور واحد كل N نبضة ساعة، لذا يكون تردد إشارة الخرج مساوياً لتردد ساعة الدخل مقسوماً على N.

وفي حال انخفاض إشارة التأهيل GATE إلى الصفر أثناء العد، يتوقف العد، وعند عودة إشارة GATE إلى الواحد يعود العداد إلى شحن القيمة الموجودة في سجل العداد من جديد عند نبضة الساعة التالية، ويعود إلى العد تنازلياً من تلك القيمة. أما إذا كتبت قيمة عد جديدة في سجل العداد، فلن تنتقل القيمة الجديدة إلى العداد حتى تصل قيمة العد إلى الواحد.

النمط 3: الموجة المربعة Square Wave

إذا بُرمج أحد العدادات للعمل في النمط 3 وشُحنت في سجل العداد قيمة زوجية، يعطي العداد على خرجه إشارة مربعة ترددها يساوي تردد ساعة الدخل مقسوماً على القيمة المشحونة في العداد. أما إذا شحنت فيه قيمة فردية فلن تكون إشارة الخرج متناظرة، أي قد تكون المدة التي تبقى النبضة فيها في المستوى العالي أكبر من مدة المستوى المنخفض بدور ساعة دخل أو أكثر.

بعد شحن العداد بالقيمة المطلوبة تنتقل القيمة المشحونة في سجل العداد إلى العداد نفسه، ثُم يبدأ العد التنازلي بمقدار عدتين، وعند وصول قيمة العداد إلى القيمة 2 ينتقل خرج العداد من الواحد إلى الصفر، ويعاد شحن قيمة العد إلى العداد آلياً. يبقى خرج العداد على الصفر حتى وصول العد ثانية إلى القيمة 2، فيعود الخرج إلى الواحد ويعاد شحن العداد بقيمته الأولية وهكذا.

في حال انخفاض المدخل GATE في أية لحظة، يتوقف العد، وتعود قيمة العد المختزنة في سجل العداد للشحن في العداد فور عودة إشارة GATE إلى الواحد.

النمط 4: القدح البرمجي Software-triggered Strobe

في هذا النمط يقوم العداد بتوليد نبضة قدح وحيدة تنتقل من الواحد إلى الصفر بعد N+1 نبضة ساعة من شحن العداد بالقيمة N، تستمر مدة دور واحد من ساعة الدخل. وقد سمي هذا النمط بالقدح البرمجي لأن النبضة تتولد نتيجة كتابة قيمة العد في العداد بتعليمة برمجية. وبعد وصول العد إلى الصفر ينتقل العداد إلى FFFFh ثم يتابع العد تنازلياً.

يمكن استغلال هذا النمط لكتابة معطيات تفرعية على معبر ما، ثم إرسال نبضة قدح بعد تأخير ما، لإعلام الجهاز المستقبل بجاهزية المعطيات.

النمط 5: القدح المادي Hardware-triggered Strobe

يستخدم هذا النمط عند الحاجة إلى توليد نبضة قدح تنتقل من الواحد إلى الصفر، بعد تأخير معين من انتقال إشارة الدخل GATE للعداد من الصفر إلى الواحد. ويفيد هذا النمط بتأخير النبضة الصاعدة الواردة إلى المدخل GATE بقدر معين.

يُشحن سجل العداد بقيمة العد المطلوبة، التي تنتقل إلى العداد فور انتقال إشارة دخل العداد GATE إلى الواحد. يبدأ العداد بالعد

تنازلياً بعد انتقال قيمة العد إلى العداد بنبضة ساعة دخل واحدة. عندما يصل العداد إلى الصفر، ينتقل خرج العداد إلى الصفر مدةً تساوي نبضة ساعة دخل وحيدة. أي إن الخرج ينتقل إلى الصفر بعد N+1 نبضة من انتقال مدخل القدح GATE إلى الواحد.

في حال ورود نبضة قدح ثانية إلى المدخل GATE أثناء العد، يعود العداد إلى شحن قيمة العد الأولية، ويبدأ العد التنازلي من جديد. وفي حال استمرار ورود نبضات القدح قبل وصول العداد إلى القيمة صفر يبقى الخرج OUT على المستوى العالي.

أما إذا كتبت قيمة عد جديدة في سجل العداد أثناء العد، فلن تنتقل القيمة الجديدة العداد حتى ورود إشارة قدح إلى المدخل GATE. يمكن العودة إلى تفاصيل أكثر، وأشكال الإشارات الموافقة، في نشرة معلومات المؤقت/العداد 8254.

الفصل الثالث

مدخل إلى البرمجة بلغة المجمّع

1 مقدمة

يهدف هذا الفصل إلى عرض أساسيات البرمجة بلغة المجمّع (أو لغة التجميع) Assembly Language. نعرض أولاً أنماط العنونة المختلفة، التي تسمح للمعالج بالوصول إلى المعطيات في الذاكرة، ثم نتطرق بعد ذلك إلى المراحل الأساسية لتطوير برنامج بلغة المجمّع، بدءاً من تعريف المسألة وحتى كتابة البرنامج.

2 لغات البرمجة

يقوم أي معالج صغري، مهما يكن نوعه، بتنفيذ برنامج مخزّن في الذاكرة، ولذا فإن المعالج ينفذ برنامجاً مكتوباً بالصيغة الاثنانية (1,0) فقط. وكما هو معروف، توجد ثلاثة مستويات مختلفة لكتابة برنامج لمعالج صغري، نعرضها باختصار تباعاً.

2-1 لغة الآلة

يمكن كتابة البرامج ببساطة كمتتالية أرقام اثنانية، تمثل تعليمات المعالج الواجب تنفيذها (انظر الشكل 1).

المحتوى الاثناني	المحتوى الست عشري	عنوان الذاكرة	العملية
00100 h	E4	11100100	INPUT From
00101 h	05	00000101	Port 05 h
00102 h	04	00000100	ADD
00103 h	07	00000111	07 h
00104 h	E6	11100110	OUTPUT To
00105 h	02	00000010	Port 02

الشكل 1: مثال على برنامج معالج صغري.

تسمى الصيغة الاثنانية للبرنامج، بلغة الآلة Machine Language، لأنها الشكل الوحيد الذي تفهمه الآلة (أي الحاسوب). ولكن، من الصعب، إن لم يكن من المحال، على مبرمج أن يحفظ آلاف الرموز الاثنانية لمعالج ما مثل المعالج 8086. ثم إن حدوث خطأ أثناء العمل أمر وارد جداً، إذ يكفي تبديل 0 مكان 1 أو العكس! إن استخدام الصيغة الست عشرية قد تساعد على التغلب على هذه المشكلة، فهي تمثيل أكثر تراصاً من الصيغة الاثنانية، ولكنها لا تحول دون مشكلة حفظ الآلاف من رموز التعليمات.

2-2 لغة المجمّع

لجعل البرمجة أيسر، يكتب معظم المبرمجين بلغة المجمّع، ثم يترجمون برامجهم إلى لغة الآلة، بحيث يمكن شحنها في الذاكرة وتنفيذها. تستخدم لغة المجمّع مصطلحات «تذكّرية» mnemonics ذات ثلاثة أو أربعة أحرف لتمثيل كل نوع من التعليمات. والمصطلح هو مجرد وسيلة لمساعدة المبرمج في التذكر، والحروف المستخدمة فيها تدل عادة على كلمة معينة في اللغة الإنكليزية وتشير إلى العملية التي تقوم بها التعليمة. فمثلاً، يُرمز إلى عملية الطرح بالرمز SUB (المشتق من فعل Substract في اللغة الإنكليزية)، وتدل تعليمة OR على العملية المنطقية الموافقة، وكذلك تعليمة XOR تدل

على عملية الجمع الحصري، أما تعليمة نقل معلومة ما من مكان إلى آخر فهي تعليمة MOV [1].

والسؤال الذي يطرح ذاته هنا: بفرْض أننا كتبنا برنامجنا بلغة المجمّع، كيف يمكن الانتقال إلى برنامج بلغة الآلة، حتى يستطيع المعالج الصغري تنفيذه ؟

في الواقع، هناك طريقتان. الطريقة الأولى هي الترجمة من لغة المجمّع إلى لغة الآلة يدوياً، وذلك بكتابة الصيغ الاثنانية المقابلة لكل تعليمة بلغة المجمّع. ولكن هذه الطريقة محفوفة بالمخاطر، إذ من السهل ارتكاب الخطأ أثناء الترجمة، والحصول على برنامج مخالف لما أردناه. أما الطريقة الثانية، وهي الأسلم، فهي استخدام مجمّعٍ Assembler، هدفه ترجمة البرنامج من لغة المجمّع إلى لغة الآلة. وهذا المجمّع هو برنامج يعمل على أي حاسوب شخصي، وغالباً ما يترافق مع أدوات مساعدة لتطوير البرامج: مثل المقلد المحاكي Simulator، والمنقح Debugger. ومن الجدير بالذكر، أن لكل معالج صغري لغة مجمّع خاصة به، لأن هذه اللغة تعكس مباشرة إمكانات البرمجة المتاحة لهذا المعالج.

3-2 اللغات العالية المستوى

يمكن كتابة برنامج لمعالج صغري باستخدام إحدى اللغات التي تسمى اللغات العالية المستوى High-Level Languages، مثل Fortran، أو BASIC، أو Pascal، أو C، أو Ada. تستخدم هذه اللغات عبارات برمجية قريبة من اللغة الإنكليزية «الاعتيادية». ولذا، فهي أسهل للكتابة من لغة المجمّع. وقد تمثل أي عبارة برمجية من لغة عالية المستوى عدة تعليمات من لغة الآلة. فلتحويل برنامج مكتوب بلغة عالية المستوى إلى برنامج بلغة الآلة ينبغي استخدام «المترجم» Compiler. وعند البرمجة بلغة عالية المستوى، يكون الزمن

---

1 انظر مجموعة تعليمات المعالج 8086 في الملحق الرابع.

اللازم لإنجاز البرنامج أقصر منه بلغة المجمّع، لأن اللغة العالية المستوى تستخدم لبنات بنيوية أغنى لكتابة البرنامج. ولكن هذه البرامج، بعد ترجمتها إلى لغة الآلة، تُنفذ ببطء أشد منه في حالة الكتابة مباشرة بلغة المجمّع أو لغة الآلة، لأن المترجم المستخدم للتحويل قد لا يكون أمثلَ.

ولذا فالبرامج التي تتعامل مع الكيان الصلب بشدة، كبرامج التحكم في الأذرع الآلية (الروبوتات) مثلاً، من الأفضل كتابتها مباشرة بلغة المجمّع، حتى لا يستغرق تنفيذها زمناً أطول مما يستحق. أما البرامج التي تعالج كميات هائلة من المعطيات، مثل سجلات شركات التأمين، فمن الأيسر كتابتها بلغة عالية المستوى، لأن زمن تنفيذها أمر غير حرج.

إن اتخاذ القرار باللغة الواجب استخدامها لكتابة البرنامج، أصبح أمراً محيراً حالياً، لأن المجمّعات الحديثة تسمح باستخدام بعض العبارات بلغات عالية المستوى، كما أن بعض مترجمات اللغات عالية المستوى أصبحت تتيح استخدام عبارات مكتوبة بلغة المجمّع.

في هذا الفصل، سنركّز اهتمامنا في لغة المجمّع للمعالج 8086، التي تسمح بالتعامل مع الكيان الصلب تعاملاً مباشراً، لأن هدفنا البعيد هو فهم الحاسوب الشخصي المبني على عائلة المعالجات المتوافقة مع 8086.

3 مبدأ التجزئة

لابد قبل التطرق إلى تقنيات البرمجة بلغة المجمّع من معرفة الطرائق التي يستطيع بها المعالج الوصول إلى المعطيات في الذاكرة. وقد رأينا في الفصل الأول كيف يضع المعالج 8086، عند النفاذ إلى الذاكرة، عنواناً على مسراه مرمزاً على 20 خانة. ولكن سجلات المعالج الداخلية لا تخزن إلا قيماً ذات 16 خانة، لذلك استُخدمت في المعالج

8086 (وفي الأجيال اللاحقة من عائلة معالجات Intel) طريقة تسمى التجزئة إلى قطاعات Segmentation.

لتوليد العنوان الحقيقي Physical Address المرمز على 20 خانة، يقوم المعالج بجمع قيمة قاعدية Base Address، مرمزة على 16 خانة ومخزّنة في مؤشر من مؤشراته الداخلية، إلى قيمة انزياح Offset مرمزة أيضاً على 16 خانة ومخزّنة في سجل من سجلات المعالج الداخلية كما هو موضح في الشكل 2.

نلاحظ إذن أن القاعدة تزاح نحو اليسار أربع خانات، وتملأ الخانات المزاحة بقيم صفرية ثم تجمع إلى قيمة الانزياح، التي تعد قيمة على 20 خانة، ولكن خاناتها الأربع العليا مساوية للصفر. نحصل بعد الجمع على قيمة مرمزة على 20 خانة تمثل العنوان الحقيقي الواجب وضعه على المسرى للحصول على المعلومة من الذاكرة.

تُستخدم هذه الطريقة للوصول إلى المعلومات في أي قطاع Segment من الذاكرة: قطاع المعطيات، أو قطاع البرنامج، أو قطاع المكدس، أو القطاع الإضافي. إذ تكون القيمة القاعدية اللازمة لتوليد العنوان الحقيقي مخزنة في المؤشر المقابل لكل قطاع. فمثلاً عند الوصول إلى قيمة في قطاع المعطيات، يكوّن محتوى السجل CS العنوان القاعدي، ويكون الانزياح مخزناً في أي من سجلات المعالج AX, BX, CX, DX. وعند الوصول إلى قيمة في قطاع المكدس يستخدم السجل SS كقيمة قاعدية في توليد العناوين، ويكون الانزياح مخزناً في السجلSP. وعند الوصول إلى تعليمة في قطاع البرنامج تكون القاعدة موجودة في السجل CS والانزياح في السجل IP.

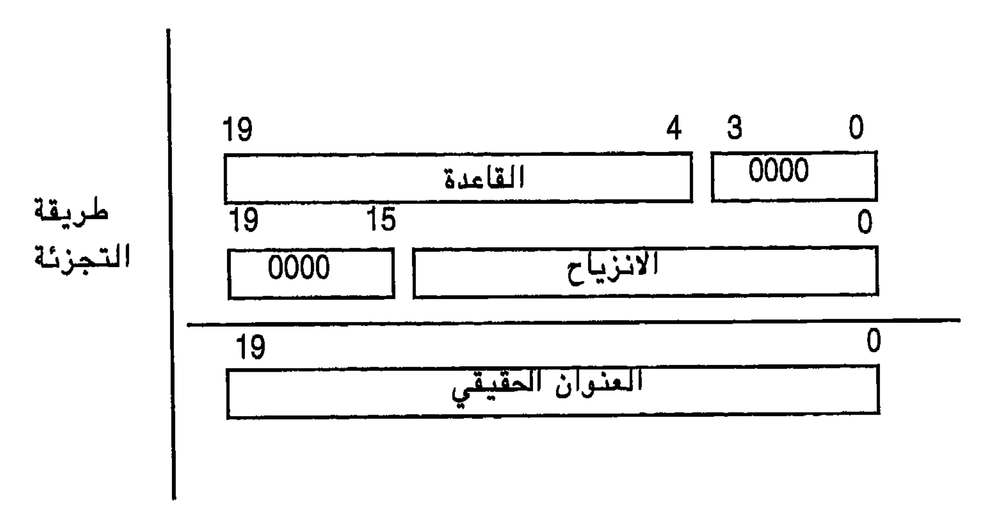
<u>مثال:</u>

نريد الوصول إلى المعلومة المخزنة في العنوان الحقيقي 2437Ah في قطاع المعطيات. بفرض أن قيمة السجل DS هي 2000h، فما هو الانزياح الواجب جمعه للوصول إلى تلك المعلومة؟

لتوليد العنوان الحقيقي 2437A ينبغي أولاً إزاحة السجل DS أربع خانات إلى اليسار، ثم جمع محتواه إلى الانزياح المناسب أي:

24 37 A	
26 00 0 –	(DS بعد الإزاحة)
04 37 A	الانزياح

و من ثم، نحصل بالطرح على قيمة الانزياح المرمز على 16 خانة.



الشكل 2: آلية حساب العنوان الحقيقي.

في الواقع، هناك طرق عديدة لتقديم هذا الانزياح إلى وحدة التنفيذ EU. فقد تحتوي التعليمة ذاتها على قيمة الانزياح، أو قد يُخزن الانزياح في سجل ما في المعالج، أو في موقع في الذاكرة. تسمى هذه الطرائق أنماط العنونة Addressing Modes، وستُعرض بالتفصيل في الفقرة التالية.

4 أنماط العنونة

1-4 العنونة الفورية

في نمط العنونة الفورية Immediate Addressing، تحتوي تعليمة المعالج على القيمة «الفورية» الواجب استخدامها أثناء التنفيذ، وبكلمة أخرى، فإن حدود التعليمة هي قيمة معطاة.

مثال 1:

```
MOV CX, 437Bh
```

ستنقل القيمة 437Bh، بعد تنفيذ هذه التعليمة، إلى السجل CX.

ملاحظة:

في التعليمة MOV، يدل الحد الأول دوماً على الوجهة، والحد الثاني على المصدر.

مثال 2:

```
MOV CL, 48 h
```

سيصبح محتوى السجل CL (المرمز على 8 خانات) 48h.

يمكن أذن نقل قيم ذات 8 خانات أو 16 خانة إلى أحد سجلات المعالج، أو أحد مواقع الذاكرة نقلاً فورياً.

2-4 العنونة بالسجل

في نمط العنونة بالسجل Register Addressing، تحتاج تعليمة المعالج إلى القيمة المخزّنة في أحد سجلاته الداخلية. وبمعنى آخر، تكون حدود التعليمة سجلات المعالج الداخلية.

<u>مثال</u> 1:

MOV CX, AX

عند تنفيذ هذه التعليمة، تُنسخ القيمة المخزنة في السجل AX والمرمزة على 16 خانة إلى السجل CX.

<u>مثال</u> 2:

MOV CL,AL

تنقل هذه التعليمة محتوى السجل AL ذا الخانات الثمان إلى السجل CL المرمز على ثمان خانات، أي يمكن بواسطة هذا النمط نقل قيم سجلات مرمزة على 16 أو 8 خانات.

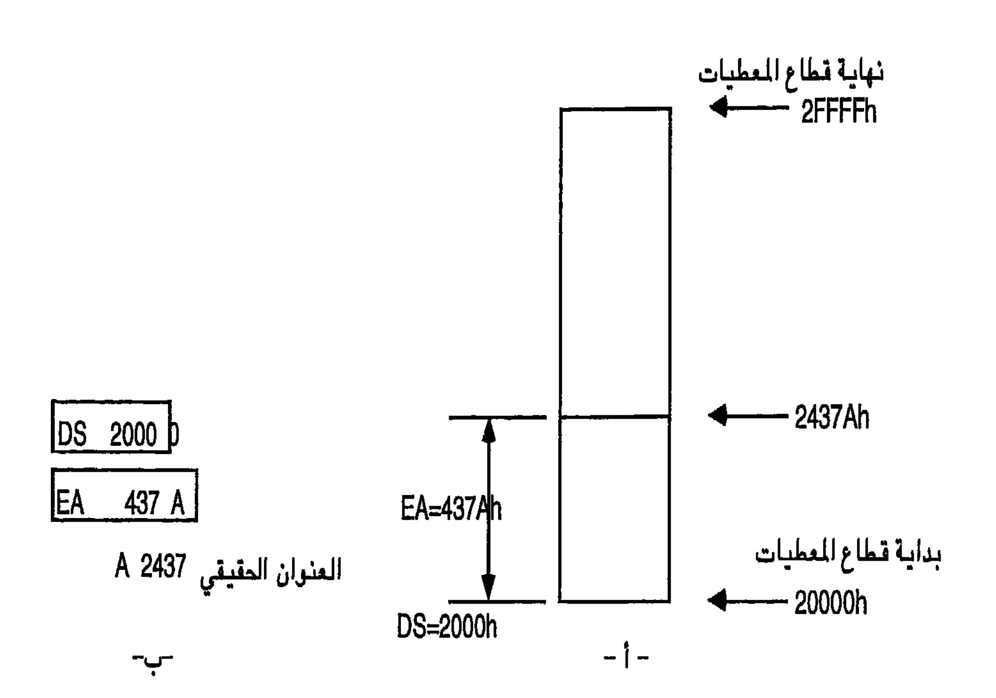
3-4 العنونة المباشرة

رأينا في العنونة الفورية أن حد التعليمة هو القيمة الواجب استخدامها. أما في نمط العنونة المباشرة Direct Addressing، فحد التعليمة هو انزياح عنوان موقع في الذاكرة، يحتوي على القيمة الواجب استخدامها. تُمثل هذه العنونة في الشكل 3.

<u>مثال</u> 1:

MOV CL, [437Ah]

عند تنفيذ هذه التعليمة، تُنقل إلى السجل CL، المرمز على 8 خانات، القيمة الموجودة في الذاكرة في العنوان 437Ah. وفي الواقع، ينتج العنوان الحقيقي الكامل لهذه القيمة (على 20 خانة) من إزاحة القاعدة المخزّنة في السجل DS أربع خانات نحو اليسار، ثم جمعها إلى الانزياح 437Ah. وبمعنى آخر، تكون القاعدة مخزّنة دوماً في السجل DS ما لم يُشر إلى عكس ذلك.



الشكل 3: العنونة المباشرة.

مثال 2:

MOV BX, [437Ah]

في هذا المثال تُنقل إلى السجل BX، ذي 16 خانة، القيمة الموجودة في موقعين متتابعين من الذاكرة كل منهما على 8 خانات؛ بحيث يكون انزياح الموقع الأول هو 437Ah، وانزياح الموقع الثاني هو 437Bh. وبكلمات أخرى، ستُنقل القيمة ذات الانزياح 437Ah إلى السجل BL، والقيمة ذات الانزياح 437Bh إلى السجل BH.

مثال 3:

MOV[437Ah], BX

تقوم هذه التعليمة بتخزين محتوى السجل BX في موقعين متتابعين من الذاكرة بالانزياح 437Ah و 437Bh تخزيناً مشابهاً لما وُصف سابقاً.

ملاحظة 1:

يجب الانتباه، عند استخدام العنونة المباشرة، إلى وجود قوسين يحيطان بقيمة الانزياح. وفي حال نسيانهما، تتحول التعليمة إلى نمط عنونة فورية، وتُنفذ، من ثَم، تنفيذاً مخالفاً لما هو متوقع. فمثلاً التعليمة:

MOV CX,437Ah

تؤدي إلى تخزين القيمة 437Ah في السجل CX، أما التعليمة:

MOV CX,[437Ah]

فتقوم بتخزين محتوى الذاكرة في السجل CX.

ملاحظة 2:

عند تنفيذ تعليمة نقل معلومات (مثل MOV) لقيم على 16 خانة في النمط المباشر، يجب التيقّن أن الانزياح هو قيمة زوجية.

4-4 العنونة غير المباشرة بالسجل

في نمط العنونة غير المباشرة بالسجل Register Indirect Addressing، يُعطى في التعليمة السجل الذي يحتوي على انزياح عنوان المعلومة في الذاكرة. وبكلمة أخرى، فإن حد التعليمة المرمزة في النمط غير المباشر سجل محتواه انزياح عنوان القيمة المطلوبة. ويوجد لهذا النمط من العنونة أربعة أشكال مختلفة:

أ العنونة غير المباشرة بالسجل بدون انزياح

وهو الشكل الأبسط، إذ يمثل محتوى السجل انزياح عنوان القيمة في قطاع المعطيات.

مثال:

MOV [BX], AX

تقوم هذه التعليمة بنقل محتوى السجل AX إلى موقعين متتاليين في الذاكرة. يتحدد العنوان الحقيقي للموقع الأول بجمع محتوى السجل BX إلى محتوى مؤشر المعطيات DS (ما لم يُشر إلى عكس ذلك صراحةً) بعد إزاحة هذا المؤشر أربع خانات نحو اليسار. وهكذا نجد أن محتوى السجل BX هو انزياح عنوان موقع التخزين الأول، وأن موقع التخزين الثاني يقابل القيمة التالية مباشرة.

ب- العنونة غير المباشرة بالسجل مع انزياح ثابت

في هذا الصنف من العنونة تُضاف قيمة ثابتة إلى محتوى السجل المذكور في التعليمة للحصول على انزياح عنوان المعلومة في الذاكرة.

مثال:

MOV SI, [BP+4]

تنقل هذه التعليمة، إلى السجل SI، القيمة ذات الانزياح المساوي لناتج جمع محتوى السجل BP والقيمة 4. ولما كان للسجل SI 16 خانة، فتُنقل إليه قيمتان متتاليتان من الذاكرة.

جـ العنونة غير المباشرة باستخدام سجل الدليل

بدلاً من إضافة قيمة ثابتة إلى محتوى السجل المستخدم في العنونة غير المباشرة، يمكن إضافة محتوى أحد سجلات الدليل (وهي: SI,DI,BI). فيصبح إذن انزياح عنوان القيمة المطلوبة مساوياً لجمع محتوى السجلين معاً: سجل الدليل وسجل العنونة.

مثال:

OR [BX+DI], 0100h

تجري هذه التعليمة العملية المنطقية OR بين القيمة 0100h وموقع في الذاكرة يُعطى انزياحه بجمع محتوى السجلين DI وBX

معاً (على 16 خانة). وكما أصبح مألوفاً، تؤخذ القيمة المخزنة في موقعين متتاليين في الذاكرة لتكوين كلمة من 16 خانة.

د العنونة غير المباشرة باستخدام سجل الدليل وانزياح ثابت

يمكن أيضاً إضافة قيمة ثابتة إلى ناتج جمع محتوى سجل الدليل وسجل العنونة لتكوين انزياح عنوان القيمة المطلوبة.

مثال:

AND DL, [BX+SI+02h]

تجري هذه التعليمة، وهي تتعامل مع قيم مرمزة على 8 خانات، عملية AND بين السجل DL وموقع محدد في الذاكرة. ويعطى انزياح هذا الموقع بجمع محتوى السجلين BX و SIإلى القيمة الثابتة 02h.

5-4 العنونة بالدليل

تعتمد التعليمات التي تستخدم نمط العنونة بالدليل Index Addressing على أحد سجلي الدليل SI وDI للحصول على انزياح عنوان القيمة المطلوبة في الذاكرة. هذا النمط إذن مشابه لنمط العنونة غير المباشرة بالسجل، ولكن الاختلاف يكمن في السجل المستخدم. فهنا لا نستخدم إلا السجلين SI أو DI، أما في ذاك النمط، فيمكن استخدام سجلات المعالج AX,BX,CX,DX للعنونة.

مثال:

ADD AX,[SI]

يضاف محتوى السجل SI إلى مؤشر قطاع المعطيات DS للحصول على العنوان الحقيقي الكامل على 20 خانة. تؤخذ القيمة الموجودة في ذلك العنوان وفي العنوان الذي يليه مباشرة لتُجمع إلى السجل AX، وتُخزّن النتيجة في السجل AX.

5 مراحل تطوير برنامج بلغة المجمّع

عندما يُطلب كتابة برنامج بلغة المجمّع، ينبغي اتباع المراحل الأربع التالية لضمان بناء برنامج صحيح البنية، قادر على أداء المهمة المطلوبة.

1-5 تعريف المسألة

الخطوة الأولى في كتابة برنامج ما، هي التفكير ملياً في المسألة المراد حلها. وبكلمات أخرى، ينبغي أن نسأل أنفسنا مرات عدة «ماذا أريد أن يفعل هذا البرنامج؟!» فإننا إن لم نفعل ذلك، فقد نكتب برنامجاً يعمل، وقد يكون ضخماً جداً، ولكنه لا يحقق الغرض الذي كُتب لأجله. وعند إمعان النظر في المسألة، يجب أن نكتب واجبات البرنامج بالتسلسل الذي ينبغي اتباعه. وفي هذه المرحلة، نحن لا نكتب برنامجاً، باستخدام العبارات البرمجية الخاصة بلغة المجمّع، ولكننا نعبر عن مهمات البرنامج Tasks بكلمات «اعتيادية». فمثلاً، قد تصاغ مسألة برمجة على النحو الآتي:

1. قراءة درجة الحرارة من محس معين.
2. إضافة معامل تصحيح قدره 7.
3. تخزين النتيجة في الذاكرة.

ففي هذا البرنامج، هناك ثلاث مهمات مطلوبة، وهي مَصوغة بلغه تشبه لغة المجمّع. أما في المسائل التي هي أكثر تعقيداً، فيحسن بنا تجزئة المهمات الصعبة إلى مهمات فرعية Sub-task أسهل، وكتابة مراحل تنفيذ كل من هذه المهمات الفرعية.

2-5 تمثيل عمليات البرنامج

تُسمى سلسلة العمليات المستخدمة في حل مسألة برمجة:

خوارزمية البرنامج Algorithm. وفي هذه الفقرة، نعرض بعض الطرائق المستخدمة في تمثيل خوارزمية برنامج ما.

1-2-5 قوائم المهمات التتابعية

على نحو مشابه لما عُرض في الفقرة السابقة، يضع بعض المبرمجين قائمة بالمهمات المطلوبة، تسمى بقائمة المهمات التتابعية Sequential Task Lists، وذلك بغية توضيح خوارزمية البرنامج. لنفترض، على سبيل الإيضاح، أننا نريد، بدلاً من أخذ عينة واحدة من محس الحرارة، أن نأخذ عينة كل ساعة من ساعات اليوم وذلك خلال يوم كامل، وإضافة 7 إلى كل منها، ثم تخزين القيمة المصحّحة في الذاكرة. يمكن صياغة قائمة مهمات هذا البرنامج كما يلي:

1 قراءة عينة من محس الحرارة.
2 إضافة 7 إلى القيمة المقروءة.
3 تخزين القيمة المصحّحة في موقع من الذاكرة.
4 انتظار ساعة كاملة.
5 قراءة عينة أخرى من محس الحرارة.
6 إضافة 7 إلى القيمة المقروءة.
7 تخزين النتيجة في الذاكرة.
...
97 قراءة عينة من محس الحرارة.
98 إضافة 7 إلى القيمة المقروءة.
99 تخزين النتيجة في الذاكرة.

نلاحظ مما سبق أن هذا النهج ليس فعالاً في تمثيل عمليات البرنامج، فقائمة المهمات أصبحت طويلة جداً. ولذا، يمكن تمثيل عمليات البرنامج بالصيغة المضغوطة التالية.

- قراءة عينة من محس الحرارة.
- إضافة 7 إلى القيمة المقروءة.

- تخزين القيمة الملحقة في الذاكرة.
- انتظار ساعة كاملة.
- هل أصبح لدينا 24 عينة؟
  - إذا كان الجواب كلا : عُد إلى المهمة الأولى.
  - إذا كان الجواب نعم : توقف عن العمل.

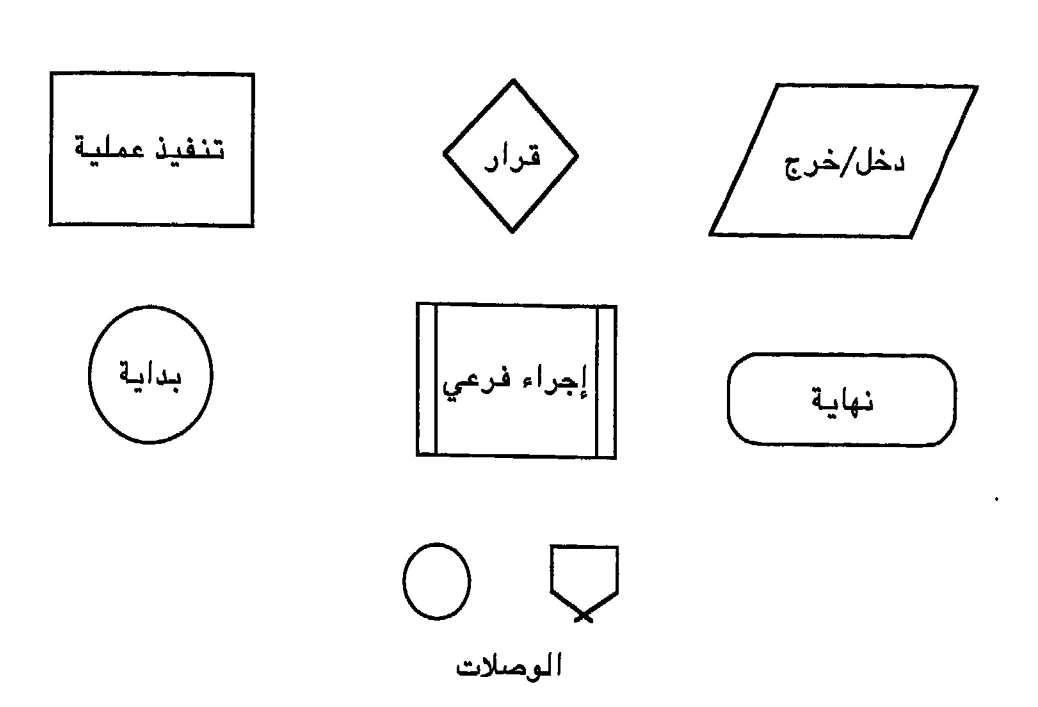
تدل الأسطر الثلاثة الأخيرة على أننا نريد تكرار عمليات القراءة والتصحيح والتخزين والانتظار 24 مرة. ومن الجدير بالذكر، أن كتابة قوائم المهمات جيداً يسهل أمر تحويلها إلى لغة المجمّع. إذ يكفي لإجراء ذلك، تحديدُ بعض التفاصيل المتعلقة بالمكونات المادية، مثل عنوان معبر القراءة، ومواقع التخزين في الذاكرة.

2-2-5 المخططات التدفقية

المخططات التدفقية Flowcharts أشكال بيانية تمثل مختلف عمليات البرنامج، فيُرمز إلى كل عملية برمز بياني. ويظهر في الشكل 4 بعض الرموز البيانية المستخدمة في بناء المخططات التدفقية.

يمكن رسم المخطط التدفقي للمثال السابق كما في الشكل 5، إذ يدل الشكل الإهليلجي على بداية البرنامج أو نهايته، كما يُرمز إلى عمليات القراءة/الكتابة من الدخل/الخرج بمتوازي الأضلاع. أما العمليات الحسابية والمنطقية ونقل المعلومات فيُرمز إليها بمستطيل.

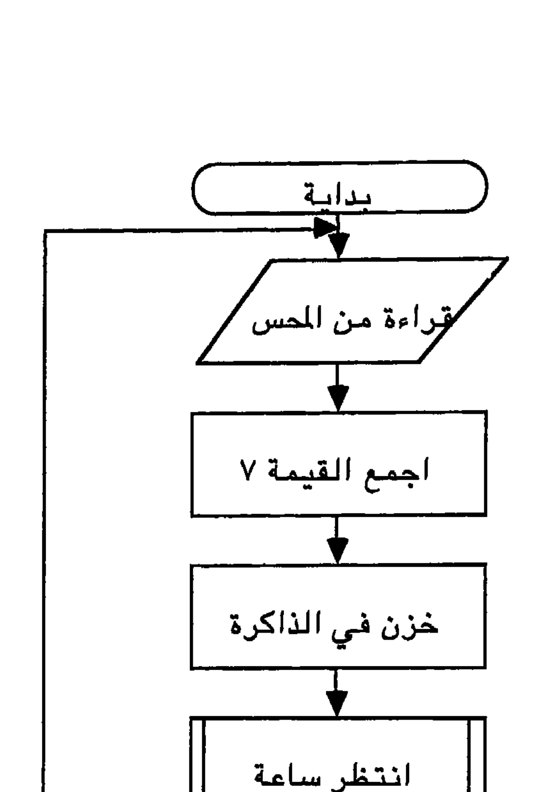
يُرمز إلى البرامج الفرعية التي تحقق وظيفة محددة بمستطيل مخطط. ويفضل دوماً الاستعاضة عن مجموعة التعليمات التي تهدف إلى تحقيق وظيفة محددة ببرنامج فرعي. فمثلاً، إذا كان لدينا مجموعة من التعليمات لحساب الجذر التربيعي في مسألة معينة، فمن الأسهل وضعها في برنامج فرعي، ثم طلب هذا البرنامج باسمه عند الحاجة إلى حساب الجذر التربيعي، فهذا أيسر من كتابة التعليمات اللازمة لذلك في كل مرة.



الشكل 4: الرموز المستخدمة في المخططات التدفقية.

أما اتخاذ القرار -الاختبار- فيُرمز إليه بمُعَّين، وهو يدل على شرط يُختبر في البرنامج، وتكون نتيجته صحيحة أو خاطئة. ولذا يخرج من هذا المعين خطان يتجه أحدهما نحو سلسلة التعليمات الواجب تنفيذها في حالة تحقق الشرط، والآخر يدل على ما يجري في حالة عدم تحققه.

وهناك أيضاً شكلان يدلان على «الوصلة». فإذا اضطررنا إلى كتابة مخطط تدفقي طويل ووصلنا إلى نهاية الصفحة، فيمكننا إنهاء المخطط على الصفحة برمز الوصلة الخماسية الأضلاع، ومتابعة المخطط على الصفحة التالية بالرمز ذاته. أما في حالة متابعة المخطط على الصفحة ذاتها، ولكن في عمود مختلف، فيمكن استخدام رمز الدائرة (الوصلة) لإنهاء المخطط في عمود من الصفحة، ثم متابعته في عمود آخر من الصفحة ذاتها.



الشكل 5: المخطط التدفقي لبرنامج قراءة الحرارة.

إن المخططات التدفقية وسيلة بيانية لعرض تسلسل البرنامج، إلا أنها تعاني بعض المساوئ. إذ لا يمكن كتابة إلا القدر اليسير من المعلومات داخل كل رمز. وقد تصبح المخططات معقدة إذا كان البرنامج طويلاً، وهذا ما يجعل تتبعها أمراً صعباً. نصف فيما يلي طريقة أكثر تراصاً لتمثيل الخوارزميات.

3-2-5 البنية البرمجية القياسية

بدا واضحاً للمبرمجين المحترفين، منذ بداية السبعينيات، أن نجاح أي مشروع برمجي مرهون باتباع طريقة منهجية في البرمجة. وإحدى الطرائق المنهجية هي النهج التنازلي Top-down approach. ففي هذا النهج، تُجزأ أولاً مسألة البرمجة الكبيرة إلى أقسام رئيسية Modules بحيث يُظهِر المستوى الأعلى العلاقة بين هذه الأقسام ووظائف كل منها. ويعطي هذا المستوى -بصفحة واحدة- نظرة عامة إلى كامل البرنامج. ثم يجزأ كل قسم بدوره إلى أقسام فرعية أصغر. وتستمر التجزئة إلى أن تصبح خطوات كل قسم قابلة للفهم وواضحة. وعندئذٍ، يمكن أن يُوزع العمل البرمجي على أعضاء فريق العمل للتنفيذ.

ساهم في تطوير أدوات البرمجة القياسية Standard Programming اكتشاف أن عمليات أي برنامج تُمثل بثلاثة أنواع من العمليات:

- التتابع Sequence: ويشير إلى مجموعة أعمال تجري على التعاقب.
- القرار أو الانتقاء Selection/Decision: ويعني اختيار مهمة من عدة مهمات ممكنة.
- التكرار Iteration: وتعني تكرار مجموعة مهمات إلى أن يتحقق شرط ما.

وفي الواقع، تكفي هذه العبارات الثلاث لتمثيل أي مهمة برمجية، وتسمح بالحصول على برامج واضحة. ويُظهِر الشكل 6 تمثيلاً بيانياً لهذه العبارات الثلاث.

يمكن أيضاً التعبير عن هذه الأنواع الثلاثة بلغة تقترب من اللغة المستخدمة تسمى الترميز الكاذب (Pseudo-Code). ونجد في الشكل 6 مثالاً على هذا الترميز.

إن البنية الموضحة في الشكل 6 مثال على تتابع بسيط. ففي هذه البنية، تُكتب المهمات بالتسلسل المطلوب.

مثال:

- اقرأ درجة الحرارة؛
- اجمع التصحيح 7+؛
- خزّن النتيجة؛

تمثل البنية "إذا - افعل - وإلا" عملية قرار، لأنها تسمح باختيار إحدى مهمتين تبعاً للشرط المستخدم.

مثال:

إذا كانت درجة الحرارة أصغر من 70

افعل

شغّل المسخّن؛

وإلا

أطفئ المسخّن؛

ففي هذا المثال تسمح تلك البنية بتشغيل المسخن أو إطفائه بحسب درجة الحرارة.

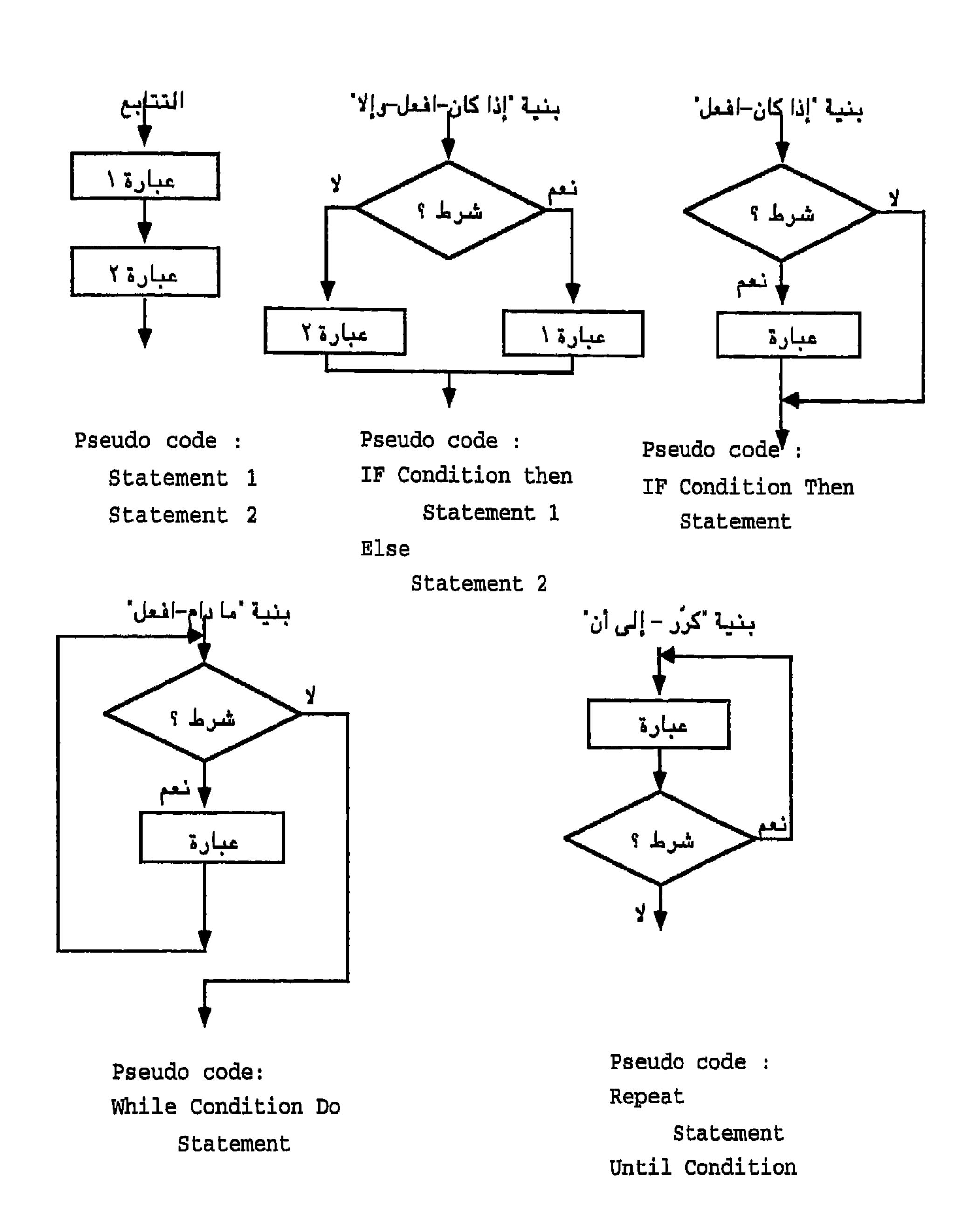
وتُعدّ عبارة "إذا - افعل" مشابهة لها، ولكنها تختلف عنها بأن أحد الممرين لا يضم مهمات للتنفيذ.

مثال:

إذا كنت جائعاً افعل

كُلْ؛

فهذه البنية لا توضح الفعل الواجب إجراؤه إذا كنت غير جائع!



الشكل 6: التمثيل البياني للعبارات القياسية.

وتعد عبارة "مادام - افعل" مثالاً على التكرار، فهي تُستخدم للدلالة على ضرورة تكرار مهمة (أو مجموعة مهمات) مادام الشرط محققاً.

مثال:

مادام هناك مال افعل

تناول العشاء خارج المنزل؛

اذهب إلى السينما؛

استقل سيارة الأجرة للعودة إلى المنزل؛

ستُكرَّر المهمات المذكورة سابقاً ما دُمتَ تملك النقود. ولاحظ أن الشرط يُختبر أولاً قبل تنفيذ المهمات.

هناك عبارة تكرار أخرى وهي: "كرّر - إلى أن"، وهي تكرّر مجموعة المهمات إلى أن يصبح الشرط محققاً.

مثال:

كرّر

اقرأ عينة من المحس؛

اجمع التصحيح 7+؛

خزّن النتيجة في موقع من الذاكرة؛

انتظر ساعة؛

إلى أن يصبح عدد العينات يساوي 24؛

نلاحظ من هذه العبارة أن المهمات تُنفذ أولاً، ثم يختبر الشرط ثانياً. ويمكن التعبير عن هذا المثال بعبارة "مادام - افعل" كما يلي:

مادام عدد العينات لا يساوي 24 افعل

اقرأ عينة من المحس؛

اجمع التصحيح 7+؛

خزّن النتيجة في موقع من الذاكرة؛

انتظر ساعة؛

إن العبارتين السابقتين تضمان عملية اختيار بسيطة من نمط "إذا كان – افعل – وإلا ". و لما كان القرار مضمّناً في هاتين البنيتين، فإننا لا نشير إلى القرار إشارة منفصلة عنهما.

وهناك بنية أخرى لعملية التكرار، تُستعمل بكثرة في لغات البرمجة العالية المستوى، وهي "من أجل – افعل"، ولها الصيغة التالية:

من أجل العداد=1 إلى n افعل

العبارة 1؛

العبارة 2؛

...

تُنفذ هذه العملية غالباً في لغة المجمّع بالبنية "كرّر – إلى أن".

وهناك أخيراً بنية أخرى للانتقاء المتعدد case أكثر تراصاً تُستخدم لانتقاء مهمة من مهمات عدة. ففي المثال الموضّح في الشكل 7 يتحقق الحاسوب (الطاهي) من اليوم لينتقي المهمات الملائمة. وما المهمات المذكورة إلا تتابع مهمات فرعية تُمثّل أيضاً ببنى مشابهة.

إن بنية الانتقاء المتعدد صيغة مضغوطة لسلسلة من عبارات "إذا – افعل – وإلا"، وهذا ما يوضحه المثال في الشكل 7.

سنصف في الفقرات التالية من هذا الفصل كيف يمكن بناء هذه العبارات بواسطة لغة المجمّع.

3-5 إيجاد التعليمات المناسبة

بعد وضع البنية العامة للبرنامج، تأتي مرحلة تحديد التعليمات المطلوبة لأداء كل جزء من البرنامج. وكما هو الحال عند تعلم لغة ما، فإن إتقان المعجم كاملاً لا يؤدي إلى الطلاقة في اللغة. فمن الأفضل إذن حفظ بعض الكلمات ووضعها في جمل مفيدة، ثم الانتقال إلى تعلم المزيد من الكلمات، واستخدامها في صيغ أكثر تعقيداً. وكذلك الأمر فيما يخص لغة المجمّع. فمن المفيد معرفة أنواع التعليمات الممكن استخدامها، ثم العودة إلى لائحة مجموعة تعليمات المعالج (الملحق 4) لتحديد طريقة الاستخدام السليم.

لنبحث على سبيل المثال عن التعليمات المناسبة لصياغة برنامجنا البسيط (قراءة المحس، والتصحيح، والتخزين). نجد من استعراض تعليمات الدخل والخرج، أن تعليمة IN تسمح بقراءة قيمة من معبر دخل، ويمكن استخدام تعليمة ADD لجمع معامل التصحيح 7 إلى القيمة المقروءة، كما يمكن استخدام التعليمة MOV لنسخ ناتج الجمع إلى مكان ما في الذاكرة. والملاحظ هنا أن تقسيم البرنامج إلى أجزاء بسيطة قد يسّر أمر إيجاد التعليمة التي تؤدي المهمة المطلوبة.

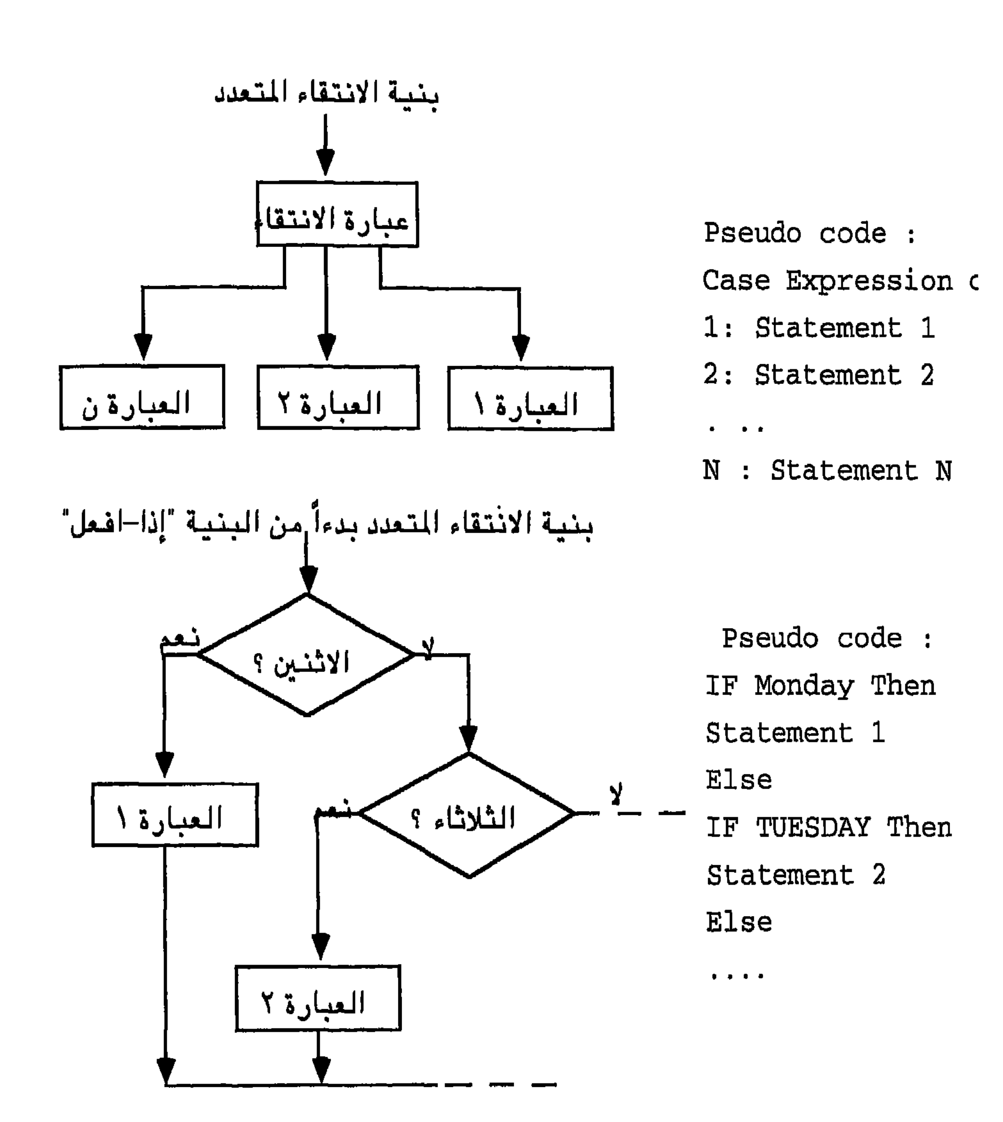
نصِف في الفقرة التالية طريقة وضع هذه التعليمات في برنامج واحد.

4-5 كتابة البرنامج

1-4-5 تعليمات الاستهلال

بعد إيجاد التعليمات اللازمة لأداء المهمات المطلوبة، لابد من البحث عن بعض التعليمات الضرورية قبل كتابة البرنامج. إن الهدف من هذه التعليمات الإضافية وضع قيمة ابتدائية في الأجزاء المختلفة من النظام، مثل سجلات القطاعات، والرايات، والطرفيات

القابلة للبرمجة. تسمّى هذه التعليمات بتعليمات الاستهلال Initialisation.



الشكل 7: التمثيل البياني لعبارة الانتقاء.

وعلى سبيل المثال، ينبغي وضع قيمة ابتدائية في سجلات القطاعات بالأوزان العليا (المرمزة على 16خانة) لعنوان الموقع الذي يبدأ منه ذلك القطاع. ففي مثالنا، نحتاج إلى وضع قيمة ابتدائية في سجل قطاع المعطيات حتى نستطيع تخزين النتيجة المصححة في موقع منه. فإذا أردنا مثلاً التخزين في الموقع ذي العنوان 00100 XXXh، كان من الضروري تخزين القيمة 0010h في سجل قطاع المعطيات. ومن أجل ذلك، يجب تخزين هذه القيمة أولاً في أحد سجلات المعالج، ثم نقلها إلى سجل قطاع المعطيات.

وإذا استعمل البرنامج المكدس، يجب شحن مؤشر المكدس بانزياح قمة المكدس.

من جهة أخرى، تضم معظم النظم الحاسوبية طرفيات متعددة قابلة للبرمجة، مثل المؤقتات، والمتحكمات. لذا، ينبغي إدراج تعليمات استهلال هذه الطرفيات في بداية البرنامج لتحديد أنماط عملها. وإضافة إلى ذلك، ينبغي من ناحية أخرى استعمال تعليمات الاستهلال لمسح رايات التحكم في المعالج (أو تأهيلها)، مثل راية تأهيل المقاطعات وراية الاتجاه.

أنسب طريقة لإعداد الاستهلال هي وضع قائمة بكافة السجلات والطرفيات المبرمجة، والرايات التي يضمها النظام الحاسوبي المراد برمجته. بعدئذٍ، يمكن أن نضع علامة عند تلك السجلات أو الرايات المستخدمة في البرنامج، ومن ثَم، إضافة تعليمات الاستهلال إلى بداية البرنامج.

2-4-5 قائمة الاستهلال

هذه قائمة بأهم المكوّنات/العناصر التي يجب التفكير فيها فيما يخص الاستهلال عند كتابة البرنامج:

- سجل قطاع المعطيات
- سجل قطاع المكدس
- سجل القطاع الإضافي

- سجل مؤشر المكدس
- سجل مؤشر القاعدة
- سجل دليل المصدر
- سجل دليل الوجهة
- البوابات المبرمجة للدارة 8255
- متحكم المقاطعات المبرمج 8259A
- العداد المُبَرمج 8254
- متحكم الاتصال التسلسلي 8251A
- متحولات المعطيات
- تأهيل/مسح راية الاتجاه، وراية تأهيل المقاطعات.

وهكذا، نجد أن القائمة قد تطول عند وضع كل الطرفيات المستخدمة في النظام، ونلاحظ أننا لم نذكر سجل قطاع البرنامج لأنه يشحن آلياً من نظام التشغيل.

3-4-5 الصيغة القياسية للبرنامج

يتضمن كل سطر من البرنامج المكتوب بلغة المجمّع الحقول التالية:

- اللصاقة Label: وهي اسم يرمز إلى نقطة معينة في البرنامج. وهو يغني عن استخدام العنوان الحقيقي لذلك السطر.

مثال:

```
PROG_Start: MOV AL, 10h
            ...
            JMP PROG_Start
```

يمكن للصاقة PROG_Start أن تحل مكان عنوان السطر الذي يدل على بداية البرنامج. ولذلك، عندما نريد القفز إلى بداية البرنامج، يكفي استخدام الاسم PROG_Start في تعليمة القفز.

• رمز التعليمة التذكري Mnemnic: ويضم هذا الحقل اسم التعليمة المراد استخدامها، مثل MOV, ADD وغيرها...

• الحدود أو المعاملات Operands: تسمح هذه الحدود بتحديد القيم اللازمة لتنفيذ التعليمة.

• التعليق Comment: يسمح هذا الحقل، الذي يبدأ بعد علامة ";" بإعطاء توضيح حول عمل كل تعليمة في البرنامج. إن هذا الحقل مفيد جداً لأمرين:

– جعل البرنامج قابلاً للفهم من الآخرين (غير المبرمج).

– بعد مرور زمن طويل على كتابة البرنامج، قد يضطر المبرمج للعودة ثانيةً إلى البرنامج لتعديله أو لتطويره. في هذه الحالة، فإن حقل التعليق ينعش ذاكرة المبرمج، ويسمح له بتذكر خطوات البرنامج التفصيلية، وهذا ما يسهل عملية التعديل أو التطوير.

ملاحظة:

من الضروري عند كتابة التعليق شرح دور التعليمة في البرنامج، لا عملها الوظيفي. فمثلاً، لا يكون التعليق على تعليمة MOV AX, 0h مثل «تعليمة نقل المعلومات»، وإنما ينبغي شرح سبب وضع القيمة 0 في السجل AX في البرنامج.

مثال:

لنكتب برنامجاً يقوم بالمهمات التالية:

– قراءة قيمة المحس.

– تصحيح القيمة بإضافة القيمة 7.

– تخزين النتيجة في الذاكرة.

```
; Programmer: NA
; Program Title: Read Temperature
; File Name: TEMP.ASM
; Description:    The program reads temperature, corrects it, and stores
;                 results in memory
; Procedures: X
; Registers used: AX
; Flags affected: All conditional
; Ports: 05h
; Memory:    0100h - Data
;            0200h -0020Ch - Code

PROG_START:   MOV AX, 0010h      ; initialize DS to point to start of
                                 ; memory set for data storage
              MOV DS, AX
              IN AL,05h          ; read temperature from port 5h
              ADD AL, 07h        ; add correction factor of +7h
              MOV [0000], AL     ; store result in memory
              INT 3              ; wait for command from user
```

ملاحظة:

إن التعليمة INT 3 تُستخدم عادة كنقطة توقف Break point؛ فهي تعيد السيطرة إلى نظام التشغيل. وبدون مثل هذه التعليمة في نهاية البرنامج، يقوم المعالج بشحن وتنفيذ تعليمة أخرى من قطاع البرنامج قد تكون بلا معنى، ومن ثَم قد تؤدي إلى تعطل كامل النظام.

4-4-5 التوثيق

أكّدنا في الفقرات السابقة ضرورة كتابة البرنامج بعناية من حيث تعريف المهمات المطلوبة، وتمثيلها بيانياً، وكتابة الخوارزمية، وصياغة البرنامج. وهنا نؤكد ضرورة توثيق البرنامج بحيث يصبح عمل البرنامج واضحاً للمبرمج، كما هو واضح لأي شخص آخر يقرؤه. يشمل التوثيق تحديد اسم البرنامج واسم الملف المحتوي عليه، وتاريخ كتابته، ورقم الإصدار(إن كان من الممكن وجود عدة إصدارات منه)،

ووَصْف موجز له، وتحديد الإجراءات المستخدمة، والبوابات الخارجية، والسجلات والرايات التي يؤثر فيها لجعل التواصل بينه وبين برامج أخرى أمراً سهلاً. وتوضع هذه المعلومات في بداية البرنامج. ويشمل التوثيق أيضاً وضع التعليقات المناسبة لكل (أو لمعظم) أسطر البرنامج لتحديد الغرض منها. ولا يُقصد بالتعليق هنا شرح عمل كل تعليمة على حدة (كما ذكرنا سابقاً)، وإنما، ينبغي شرح الغرض من استخدام تلك التعليمة في البرنامج.

إن تأكيد ضرورة التوثيق أمر مهم للغاية. إذ تدل التجارب أن برنامجاً صغيراً كُتب منذ بضعة أشهر بلا تعليقات أو شرح، يكون غير مقروء اليوم، ولو عاد إليه المبرمج ذاته!



الفصل الرابع

تقنيات البرمجة بلغة المجمّع

1 مقدمة

يهدف هذا الفصل إلى منح الطالب القدرة على برمجة المعالج 8086، وذلك بتقديم تقنيات البرمجة الأساسية بلغة المجمّع. نعرض تباعاً عمليات القفز والحلقات والتنفيذ المشروط؛ ثم نناقش البرامج الفرعية والتعليمات الموسّعة؛ وأخيراً نصف طريقة تبادل المعلومات الأساسية بين المعالج والطرفيات بواسطة المقاطعة.

2 الرايات وعمليات القفز

تأتي قوة الحاسوب الحقيقية من قدرته على تكرار مجموعة تعليمات مادام هناك شرط معين محقق، أو إلى أن يتحقق شرط ما؛ ومن قدرته على اتخاذ القرار في تنفيذ بعض المهمات دون غيرها بناء على شروط محددة.

تدل الرايات Flags على وجود شرط ما أو عدمه. وتُستخدم تعليمات القفز Jump في إعلام الحاسوب بالمهمات الواجب تنفيذها بناء على الرايات. سنناقش في هذه الفقرة رايات الشروط في المعالج 8086 وتعليمات القفز، ونعرض طريقة تنفيذ البنية "مادام – افعل".

1-2 الرايات المشروطة

يوجد في المعالج 8086 ست رايات مشروطة Conditional Flags وهي: راية الحمل، وراية التثبت، وراية الحمل المساعد، وراية الصفر، وراية الفائض.

1-1-2 راية الحمل

تتأثر راية الحمل Carry بتعليمات الجمع والطرح والمقارنة. فإذا كان مجموع عددين أكبر من أن يرمّز على 8 خانات، أصبحت راية الحمل مساوية للواحد للدلالة على وجود حمل إلى الخانة اللاحقة.

وبالمشابهة، تصبح راية الحمل مساوية للواحد عندما يكون ناتج جمع عددين مرمزين على 16 خانة أكبر من 16 خانة. أما في تعليمات الطرح، فتدل هذه الراية على الاستعارة Borrow. فإذا كان العدد المطروح أكبر من العدد المطروح منه رُفعت الراية للدلالة على ضرورة الاستعارة لإجراء الطرح.

وتؤثر أيضاً تعليمات المقارنة في راية الحمل، ذلك أن هذه التعليمات تقوم بطرح العدد المحدد بالمصدر من الوجهة، ولكن دون أن تخزن النتيجة في الوجهة. فتعكس الرايات عندئذٍ نتيجة المقارنة، فإذا كان المصدر أكبر من الوجهة، رُفعت راية الحمل/الاستعارة، وإذا كان المصدر أصغر من الوجهة بقيت راية الحمل كما هي، وفي حال التساوي، تُرفع راية الصفر للدلالة على ذلك.

مثال:

CMP BX, CX

الحالة 1: BX < CX ⟵ CF = 1, ZF =0

الحالة 2: BX > CX ⟵ CF =0, ZF = 0

الحالة 3: BX = CX ⟵ CF = 0, ZF = 1

2-1-2 راية التثبت

تُستخدم راية التثبت Parity للدلالة على أن لكلمة اثنانية عدداً زوجياً أو فردياً من القيمة '1'. فإذا كان لها عدد زوجي، وصفت تلك الكلمة بأنها زوجية، وإلا فهي فردية. وفي المعالج 8086، تُرفع راية التثبت إذا كانت الثمانيّة الدنيا للوجهة زوجية. وربما كان الاستخدام الأكثر شيوعا لهذه الراية هو التحقق من سلامة وصول المعلومات عند إرسالها من حاسوب إلى آخر عبر الخطوط الهاتفية مثلاً.

3-1-2 راية الحمل المساعد

لراية الحمل المساعد Auxiliary Carry أهمية عند معالجة أرقام عشرية مرمزة ثنائياً BCD. فهي تُرفع عندما يكون مجموع الخانات الأربع الدنيا لثمانيّة مع الخانات الأربع الدنيا لثمانيّة أخرى أكبر من أربع خانات. وهي تُرفع أيضاً عند إجراء الطرح بين ثمانيّتين، وعندما يكون حاصل طرح الخانات الأربع الدنيا لثمانيّتين أكبر من أربع خانات. تدل إذن راية الحمل المساعد في هذه الحالة على ضرورة الاستعارة. وتُستخدم هذه الراية في تعليمات DAA، DAS فقط.

4-1-2 راية الصفر

تُرفع راية الصفر Zero عندما يكون ناتج العملية الحسابية أو المنطقية صفراً. فمثلاً، عند طرح عددين متساويين توضع هذه الراية على '1'. وعند إجراء عملية AND المنطقية بين عددين وتكون النتيجة صفراً، تصبح هذه الراية واحداً. وإضافة إلى تأثرها بتعليمات الجمع والطرح، والتعليمات المنطقية، فإن هذه الراية تتأثر بتعليمة المقارنة. فإذا أصبحت الراية مساوية للواحد بعد تنفيذ تعليمة المقارنة، فهذا يدل على تساوي العددين.

وهناك تعليمة أخرى تؤثر في راية الصفر وهي DEC (الإنقاص). فهذه التعليمة تنقص واحداً من العدد المحدّد بالوجهة (سجل من المعالج

أو موقع في الذاكرة). وإذا أصبح هذا العدد صفراً بعد الإنقاص رُفعت راية الصفر. وعلى سبيل المثال، لنفترض أننا نريد تكرار مجموعة مهمات 9 مرات. فلإجراء ذلك، نشحن القيمة 09h في سجل ما، ثم ننفّذ مجموعة المهمات. ننقص بعد ذلك السجل، ونفحص راية الصفر لمعرفة وصول السجل إلى القيمة 0. فإن لم تكن الراية مرفوعة، فإننا نكرّر تنفيذ المهمات من جديد، وإلا فنتوقف عن التنفيذ.

وبالطريقة ذاتها، تؤثر تعليمة INC في راية الصفر، فعندما تزاح ثمانيّة أو كلمة مؤلفة من 16 خانة وتصبح النتيجة صفراً، فإن راية الصفر تُرفع.

5-1-2 راية الإشارة

تُمثّل الأعداد الموجبة والسالبة في المعالج 8086 بطريقة الإتمام إلى العدد 2. ووفق هذه الطريقة، تدل الخانة العليا في الكلمة أو الثمانيّة على إشارة العدد، فإن احتوت صفراً كان العدد موجباً، وإن احتوت واحداً كان العدد سالباً. وتكون راية الإشارة Sign، بعد تعليمة حسابية أو منطقية، نسخة من الخانة العليا. ومن ثَم، فإذا كانت معدومة فالناتج موجب، وإذا كانت مساوية للواحد فالناتج سالب.

وتسمح هذه الراية أيضاً بمعرفة تغيّر إشارة عدد ما. فمثلاً، يؤدي إنقاص عدد مساوٍ للصفر إلى القيمة FFh. ولما كانت الخانة الأعلى في هذا العدد قد أصبحت واحداً، فإن راية الإشارة تُرفع.

6-1-2 راية الفائض

تُرفع راية الفائض Overflow عندما يفيض ناتج تعليمة حسابية (ذات إشارة) عن 16 خانة (أو عن 8 خانات إذا كانت التعليمة تتعامل مع ثمانيّات). فمثلاً، عند جمع العدد التالي (ذي الإشارة والمرمز على 8 خانات): $(+117)_{10}$ = 0111 0101، والعدد $(+55)_{10}$ = 0011 0111 نجد 1010 1100 = $(+172)_{10}$ وهي نتيجة صحيحة ولكنها لاتُرمز على

الخانات السبع المتاحة لترميز عدد ذي إشارة. ففي حالة ثمانيّة ذات إشارة، تدل الخانة العليا المساوية للواحد على عدد سالب. ولذا تُرفع هنا راية الفائض للدلالة على أن النتيجة قد فاضت وتداخلت مع خانة الإشارة.

2-2 تعليمات القفز اللامشروط

تُستخدم هذه التعليمات لإجبار المعالج على جلب تعليمات من مكان جديد في الذاكرة. ونذكّر هنا أن المعالج 8086 يولّد العنوان الحقيقي (على 20 خانة) لتعليمة ما بجمع الانزياح، المخزّن في مؤشر التعليمات، إلى القاعدة المخزّنة في سجل قطاع البرنامج (CS). تغيّر إذن تعليمات القفز العدد المخزّن في سجل مؤشر التعليمات، وهي تغيّر أحياناً في سجل القطاع. ويُقصد بالقفز اللامشروط أن المعالج سيقفز حتماً، عند تنفيذه لهذه التعليمة، إلى المكان المحدد دون أن يفحص أي شرط.

لتعليمة JMP خمسة أنواع مختلفة نذكرها تباعاً.

1-2-2 <u>القفز القريب</u>

يؤدي القفز القريب Near Jump إلى جلب تعليمة من أي موقع في قطاع البرنامج الحالي. يسمى مثل هذا القفز أيضاً «القفز داخل القطاع». وللحصول على العنوان الجديد، يجمع المعالج الانزياح المحتوى بهذه التعليمة (والمرمز على 16 خانة مع إشارة) إلى سجل مؤشر التعليمات. ويدل الانزياح ذو الإشارة المرمز على 16 خانة على أن القفز قد يحدث إلى أي موقع يبعد بـ 32767+ أو 32768- عن الموقع الحالي. ويدل الانزياح الموجب على القفز الأمامي في البرنامج، أما الانزياح السالب فهو يدل على القفز الخلفي.

2-2-2 القفز القريب القصير

وهي حالة خاصة من القفز القريب، ولكن الانزياح فيها مرمّز على 8 خانات مع إشارة. ولذا، فالقفز يكون إلى موقع لا يبعد أكثر من 127+ أو 128- عن الموقع الحالي. ومنه الاسم: القفز القريب القصير Short Near Jump.

3-2-2 القفز القريب غير المباشر

في القفز القريب غير المباشر Indirect Near Jump، يُستبدل بمحتوى مؤشر التعليمات قيمة مخزّنة في سجل ذي 16 خانة، أو في موقعين متتاليين من الذاكرة. وتعدّ هذه التعليمة أيضاً من القفز القريب، لأنها لا تغيّر عنوان القاعدة.

4-2-2 القفز البعيد

تسمح تعليمة القفز البعيد Far Jump بالقفز إلى قطاع برنامج آخر، وهذا ما يسمى أيضاً «القفز بين القطاعات» Inter-segment jump. ولإجراء ذلك، تغيّر هذه التعليمة محتوى مؤشر التعليمات وقطاع البرنامج معاً. وتحتوي التعليمة في ثمانيّتها الثانية والثالثة على الانزياح الجديد الذي يُشحن في مؤشر التعليمات، وتحتوي في ثمانيّتها الرابعة والخامسة على قيمة قطاع البرنامج الجديد.

5-2-2 القفز البعيد غير المباشر

تؤثر تعليمة القفز البعيد غير المباشر Indirect Far Jump في مؤشر التعليمات وقطاع البرنامج معاً، ولكن القيمة الجديدة تؤخذ من أربعة مواقع متتالية في الذاكرة. فالموقعان الأوليان يحددان الانزياح الذي يُشحن في مؤشر التعليمات IP، ويحدد الموقعان التاليان القيمة التي تُشحن في سجل قطاع البرنامج. وتشير التعليمة إلى عنوان الموقع الأول فقط من هذه المواقع.

مثال توضيحي:

Address			
0000	Back:	ADD AL, 03h	; add 3 to Total
0002		NOP	
3		NOP	
4		NOP	
5		JMP Back	
6		NOP	
7		NOP	

في هذا المثال، تُكرر مجموعة التعليمات NOP عدداً لانهائياً من المرات. وتشير اللصاقة Back إلى العنوان الذي نريد العودة إليه في كل مرة. فعندما يصادف المجمع تعليمة القفز، يبحث عن السطر ذي اللصاقة Back، ويحسب انزياح ذلك السطر عن تعليمة القفز. ونلاحظ ايضاً أن التعليمات المكتوبة بعد تعليمة القفز لن تُنفذ أبداً، لأن القفز إلى السطر Back قفز غير مشروط، وعندما يدخل المعالج في هذه الحلقة (Back-JMP) فلن يستطيع الخروج منها.

تُختار تعليمة القفز المناسبة بحسب قيمة الانزياح، فإذا كان الانزياح قابلاً للترميز على 4 bits. يمكن استخدام القفز القريب القصير، وإلا فيمكن استخدام القفز القريب.

لنحسب في حالة مثالنا السابق قيمة الانزياح اللازمة: عنوان السطر Back هو 0h، وعنوان تعليمة القفز هو 6h، ولكن بعد أن ينفذ المعالج تعليمة القفز، فإن مؤشر التعليمات IP يكون مساوياً لـ 8h. فلكي يعود المؤشر إلى Back يجب إضافة انزياح قدره 8h-، ومن ثم يُكتب الانزياح السالب بالإتمام إلى 2 كما يلي:

F8h = [1111 1000]

ملاحظة:

ينبغي، لحساب الانزياح، أخذ العنوان الذي يؤشر السجل IP عليه بعد تنفيذ تعليمة JMP في الحسبان، لا عنوان تلك التعليمة فحسب.

3-2 تعليمات القفز المشروط

كما ألمعنا سابقاً، يتميزالحاسوب بقدرته على اختيار أحد مسارين تبعاً لتحقق شرط معين. ويوجد في المعالج 8086 ست رايات مشروطة تُستخدم في تعليمات القفز المشروط.

نلاحظ من لائحة تعليمات المعالج 8086 أن هناك تعليمات خاصة لمعالجة الأعداد ذات الإشارة، وأخرى مختصة بالأعداد بلا إشارة. فمثلاً، العدد بلا إشارة 0110 1100 أكبر من العدد 1001 0011، في حين أن العدد ذا الإشارة 0110 1100 أصغر من العدد 00111001.

وتُسمى تعليمات القفز التي تفحص إحدى الرايات المشروطة بتعليمات القفز المشروط، لأنها تدفع المعالج إلى القفز عند تحقق شرط ما، وإلا فإن المعالج يتابع تنفيذ التعليمات تنفيذاً متسلسلاً.

مثال:

JC SAVE

إذا كانت راية الحمل مرفوعة، فالمعالج يقفز إلى السطر ذي اللصاقة SAVE، وإلا فإنه ينفذ التعليمة التي تلي القفز مباشرة.

تعدّ جميع تعليمات القفز المشروط من نمط القفز القريب والقصير. ولذا، يجب أن تقع الوجهة ضمن قطاع البرنامج ذاته، وأن تكون مسافة القفز محصورة بين 128- و 127 ثمانيّة.

في أي برنامج مكتوب بلغة المجمّع، تسبق التعليماتُ الحسابية والمنطقية وتعليمات المقارنة عادةً تعليمات القفز المشروط.

مثال:

```
CMP BL, DH
JAE Heater _off
MOV AX , 0h
```

تقارن تعليمة CMP بين محتوى السجلين BL و DH، وتؤثر في راية الحمل وراية الصفر تبعاً لنتيجة المقارنة. أما تعليمة JAE -التي تعني اقفز إذا كانت نتيجة المقارنة السابقة أكبر أو تساوي- فهي تدفع المعالج إلى القفز إلى السطر ذي اللصاقة Heater_off إذا كان محتوى BL أكبر أو مساوياً لمحتوى DH. وفي حال عدم تحقق الشرط، يتابع المعالج تنفيذ التعليمة التالية وهي MOV.

3 الحلقات

ثمة بنى مختلفة من الحلقات Loops، نعرضها فيما يلي.

1-3 حلقة "مادام – افعل"

لبنية "مادام – افعل" While - Do الصيغة التالية:

مادام الشرط محققاً افعل

تعليمة 1؛

تعليمة 2؛

...

ففي هذه البنية يُفحص الشرط قبل تنفيذ أي مهمة من المهمات.

مثال توضيحي:

1 تعريف المسألة وكتابة الخوارزمية:

لنفترض أننا، في مسألة تحكم في عملية كيميائية، نريد أن نجعل درجة حرارة المحلول مساوية لـ 100°C قبل البدء بالعمل. فإذا

كانت درجة حرارة المحلول أقل من 100°C، ينبغي تشغيل المسخّن للوصول إلى درجة الحرارة 100°C، وإذا كانت درجة حرارة المحلول أكبر أو تساوي 100°C، فنستطيع عندئذٍ الانتقال إلى خطوة أخرى في العملية الكيميائية. يمكن في هذه المسألة استخدام بنية "مادام – افعل"، لأنها تفحص الشرط قبل تنفيذ أي مهمة. يظهر الشكل 1 المخطط التدفقي لهذه الخوارزمية.

تهدف المرحلة الأولى في الخوارزمية إلى قراءة درجة حرارة المحلول. تُقارن هذه القيمة فيما بعد بالقيمة 100°C، فإذا كانت درجة حرارة المحلول مساوية أو أكبر من 100°C فلن ينفذ المعالج تعليمات هذه البنية، وسينتقل إلى التعليمة التالية وهي إيقاف المسخّن، وإلا فإنه سيشغل المسخّن ويعود إلى قراءة درجة الحرارة ومقارنتها من جديد. لن يخرج المعالج من الحلقة إلا إذا أصبحت درجة الحرارة مساوية أو أكبر من 100°C.

2 تنفيذ *الخوارزمية:*

نفترض أن محس الحرارة موصول إلى المعبر ذي العنوان FFF8h، وأن المسخّن موصول إلى المعبر FFFAh، ونفترض أن تشغيل المسخّن يقتضي كتابة القيمة '1' إلى هذا المعبر.

يُكتب البرنامج كما يلي:

```
Temp_In:     MOV DX, 0FFF8h  ; read temperature
             IN AL, DX
             CMP AL, 100     ; compare with 100°C
             JAE Heater _off
             MOV AL, 80h     ; load code to turn heater on
             MOV DX, FFFAh   ; point to output port
             OUT DX, AL
             JMP Temp_In
Heater_off:  MOV AL, 0h      ; load code to turn heater off
             MOV DX, FFFAh   ; point to out put port
             OUT DX, AL      ; turn heater off
```

بداية
قراءة درجة الحرارة
T>100 ?
نعم
لا
شغّل المسخّن
إطفاء المسخّن
A

الشكل 1: المخطط التدفقي للتحكم في درجة حرارة محلول.

نلاحظ في المثال السابق استخدام تعليمة JAE بعد تعليمة المقارنة CMP فماذا يحدث لو استخدمنا التعليمة JE بدلاً منها، وجرت المقارنة بالقيمة 101°C بدلاً من 100°C ؟

لنفترض، عند اختبار هذا الشرط للمرة الأولى، أن الحرارة لم تكن 100°C. ومن ثم، فالمعالج سيشغل المسخّن ولن يخرج في هذه الحالة من الحلقة إلى أن ينصهر المسخن !

في المثال السابق، استخدمت تعليمة JAE لتحقيق البنية "مادام – افعل"، ولكن يجدر التنبيه إلى أن جميع تعليمات القفز المشروط تعاني مشكلة مهمة؛ وهي أنها ذات عنونة قصيرة. وبكلمة أخرى، يجب أن تنحصر مسافة القفز بين 127+ أو 128- ثمانيّة بالنسبة إلى الموقع الحالي. فإذا افترضنا في مثالنا السابق أن السطر ذا اللصاقة Heater_off يبعد مسافة 220 ثمانيّة عن تعليمة القفز، فعندئذٍ ينبغي تعديل البرنامج لحل هذه المشكلة. ويمكن لذلك استخدام تعليمة JB (اقفز إذا أدنى) كما يلي:

```
Temp_In:     MOV DX, 0FFF8h
             IN AL, DX
             CMP AL, 100
             JB Heater_on
             JMP Heater_off
Heater_on:   MOV AL, 80h
             MOV DX, FFFAh
             OUT DX, AL
             JMP Temp_In
Heater_off:  MOV AL, 0h
             MOV DX, FFFAh
             OUT DX, AL
```

فعندئذٍ يقفز المعالج إلى السطر «البعيد» Heater_off بالتعليمة اللامشروطة JMP، في حين يقفز إلى السطر «القريب» Heater_on بالتعليمة المشروطة JB.

2-3 حلقة "كرر - إلى أن"

لبنية "كرر - إلى أن" Repeat - Until الصيغة التالية:

<u>كرّر</u>

تعليمة 1؛

تعليمة 2؛

...

<u>إلى أن</u> يتحقق شرط ما؛

السمة الرئيسية لهذه البنية هي تنفيذ التعليمات مرة واحدة قبل فحص الشرط، خلافاً لبنية "مادام - افعل" التي تختبر الشرط قبل تنفيذ أي تعليمة.

<u>مثال توضيحي:</u>

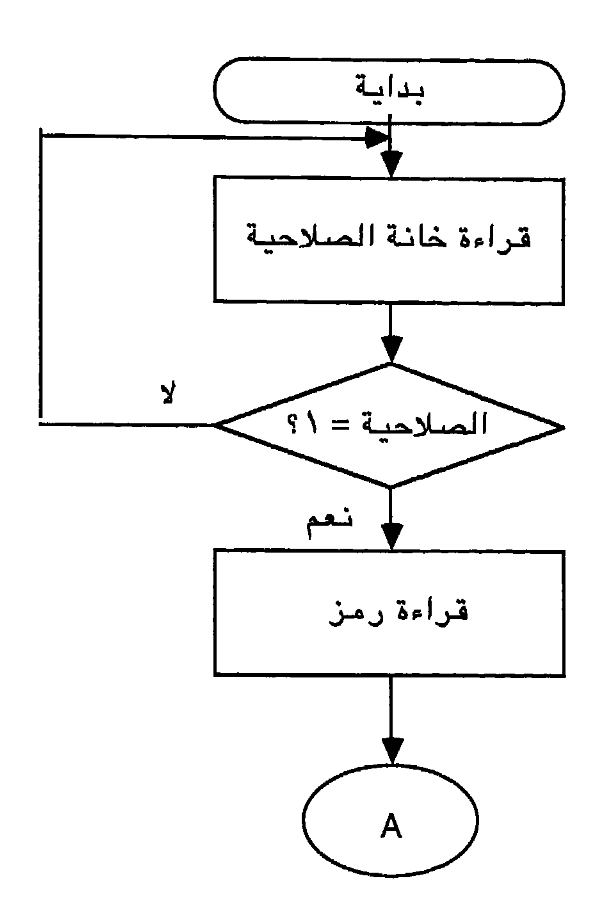
١ *تعريف المسألة وكتابة الخوارزمية:*

نريد في نظام حاسوبي ما أن نقرأ ثمانيّة من معبر معين P1 لمعرفة أن المعلومات الموجودة على المعبر الآخر P2 صالحة للقراءة. وبمعنى آخر، الكلمة الموجودة على المعبر P2 ليست ذات معنى ما لم ترد خانة الصلاحية STROBE المقروءة من المعبر P1. ولذلك، فإننا نريد انتظار خانة الصلاحية لتكون مساوية للواحد وعندها نقرأ المعلومات. يوضح الشكل 2 الخوارزمية اللازمة لهذه العملية.

2 *كتابة البرنامج:*

لنفترض أن عنوان المعبر P1 هو FFF9h، وعنوان المعبر P2 هو FFF8h.

```
            MOV DX, FFF9h    ; DX points to P1
Look_again: IN AL, AX
            AND AL, 01       ; mask extra bits and set flags
            JZ Look_again    ; if STROBE is low, keep looking
            MOV DX, FFF8h    ; DX points to P2(data port)
            IN AL, DX        ; read P2
```



الشكل 2: خوارزمية قراءة المعلومات من معبر.

نلاحظ من البرنامج السابق أننا نقرأ أولاً قيمة خانة الصلاحية من المعبر P1. ولما كنا لانهتم إلا بالخانة ذات الوزن الأدنى LSB من

الكلمة المقروءة، فإننا نعزل هذه الخانة بإجراء عملية AND المنطقية مع القيمة الثابتة 1h. وتكون نتيجة هذه التعليمة '1' إذا كانت خانة الصلاحية تساوي '1'، وإلا فإنها تكون صفراً، وفي هذه الحالة تُرفع راية الصفر ZF في المعالج. ثم نفحص راية الصفر بواسطة تعليمة القفز المشروط JE، ونقفز في حال كونها مساوية للواحد إلى السطر ذي اللصاقة Look_again، حيث نكرّر عملية القراءة من جديد.

هناك طريقة أخرى للاختبار، باستخدام تعليمة TEST التي تقوم بإجراء عملية AND المنطقية بين الثمانيّة المحددة بالمصدر والوجهة، وتؤثر في راية الصفر دون تخزين النتيجة في الوجهة.

وهناك طريقة ثالثة أيضاً لاختبار الخانة الدنيا LSB من الكلمة المقروءة، وذلك بتدويرها يميناً عبر الحمل بالتعليمة ROR. عندئذٍ، يجب استخدام تعليمة قفز مشروط لفحص راية الحمل كما يلي:

```
Look_again: IN AL, DX
            ROR AL, 1        ; rotate LSB into Carry
            JNC Look_again
```

3-3 حلقة "من أجل - افعل"

ثمة بنية مشتقة من "مادام - افعل"، ومستخدمة في لغة المجمّع، وهي "من أجل - افعل" For - Do. لهذه البنية الصيغة التالية:

من أجل العداد = 1 إلى العداد = n افعل

تعليمة 1؛

تعليمة 2؛

...

وتُنفذ هذه البنية بشحن قيمة العداد n داخل سجل من سجلات المعالج، ثم إنقاصه في كل مرة إلى أن يبلغ الصفر، فنخرج عندئذٍ من الحلقة.

مثال توضيحي:

1 *تعريف المسألة وكتابة الخوارزمية:*

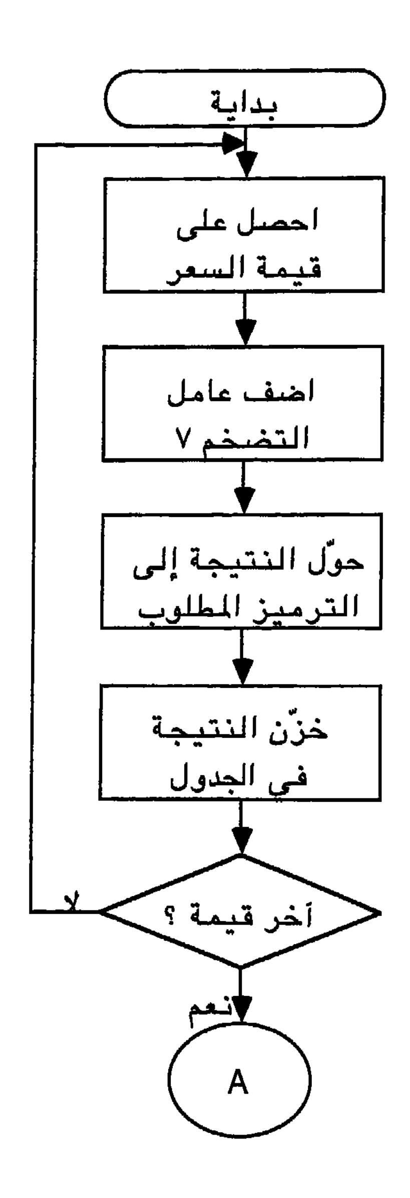
نريد قراءة ثمانٍ قيم مخزّنة في الذاكرة، وهي تمثل أسعار الكلفة لبعض المنتجات. ونريد الحصول على أسعار مبيعها التي نحصل عليها بإضافة ثابت قدره 7. ولهذا، يجب إضافة هذا العدد إلى القيمة المقروءة ثم تخزين القيم الناتجة في الذاكرة (انظر الشكل 3).

2 *كتابة البرنامج:*

تصاغ الخوارزمية السابقة بلغة المجمّع كما يلي:

```
          MOV AX, Source_data
          MOV DS, AX
          LEA BX, Prices      ; Initialize Data Segment and pointer
          MOV CX, 08h         ; Initialize Counter
Do_next:  MOV AL, [BX]
          ADD AL, 07h
          MOV [BX], AL
          INX BX
          DEC CX
          JNZ Do_ next
```

في هذا البرنامج، نهيئ أولاً سجل مؤشر قطاع المعطيات DS، ثم نشحن انزياح العنوان Prices في السجل BX، كما نشحن السجل CX بالقيمة 08h لأنه سيؤدي دور العداد في حلقة "من أجل-افعل". يُشحن بعد ذلك السجل AL بالقيمة التي يؤشر عليه السجل BX، وتُجمع هذه القيمة المقروءة إلى العدد 7، ثم تخزن في المكان ذاته بواسطة السجل BX. وأخيراً يُنقص العداد بمقدار 1 للدلالة على إنجاز حلقة في البنية "من أجل-افعل". ثم تُفحص راية الصفر، فإن لم تكن مرفوعة فهذا يعني أنه مازال هناك بعض القيم الواجب معالجتها في الذاكرة.



الشكل 3: المخطط التدفقي لخوارزمية تحديد أسعار المبيع.

4-3 تعليمات الحلقة

يمكن استخدام تعليمات الحلقة LOOP [1] في تكوين حلقات في البرنامج، فهي تتميز بإنقاص السجل CX في كل مرة وفحص قيمته (وفي بعض الأحيان تُفحص راية الصفر). فإذا كانت قيمته معدومة، فيقرر المعالج القفز إلى اللصاقة المحددة في التعليمة. ولما كان القفز هنا من النوع القصير، وجب أن تدل اللصاقة على عنوان لا يبعد أكثر من 128- أو 127+ ثمانيّة عن التعليمة.

يمكن أيضاً استخدام تعليمة JCXZ، التي تختبر قيمة السجل CX فإذا كانت معدومة قفز المعالج إلى اللصاقة المحددة بالتعليمة.

5-3 تطبيقان شهيران

1-5-3 حلقات التأخير

يُراد في بعض التطبيقات تأخير البرنامج مدة معينة من الزمن. أي إن المبرمج يرغب في تنفيذ حلقة تأخير مدة محددة. يمكن إجراء ذلك عن طريق إدراج حلقات تأخير Delay Loops في متن البرنامج. ولذلك، لابد من معرفة تردد الهزاز الخارجي الذي يحكم عمل المعالج (أو ما يسمى ساعة العمل)، وحساب زمن تنفيذ التعليمات المعطى من المصنِّع. فلكل تعليمة زمن تنفيذ محدد يُقدر بأدوارالساعة. فمثلاً، يحتاج تنفيذ تعليمة MOV إلى نقل محتوى سجل إلى آخر إلى دورَيْ ساعة. أما تعليمة JNZ فتحتاج إلى 16 دور ساعة في حال تحقق الشرط والقفز، وإلا فهي تحتاج إلى أربعة أدوار ساعة.

فإذا كانت الساعة الخارجية (تردد الهزاز) مساوية لـ 5MHz فإن دور الساعة يساوي:

$1/5MHz=0.2\mu s$

وبناء عليه، ينبغي لنا، إذا أردنا تحقيق حلقة تأخير، أن ننفذ

---

1 انظر هذه التعليمات في مجموعة تعليمات المعالج 8086 (الملحق الرابع).

مجموعة تعليمات تنفيذاً متكرراً إلى أن يمضي الوقت اللازم للتأخير.

لنأخذ مثلاً البرنامج التالي:

		عدد الأدوار اللازمة للتنفيذ	
	MOV CX, N	(T0)	4
Kill_Time:	NOP	(T1)	3
	NOP	(T2)	3
	LOOP Kill_Time	(T3)	17/5

ولنحسب زمن تنفيذ هذه الحلقة T_{all}:

$$T_{all} = T0 + N(T1+T2+T3) - 12$$
$$T_{all} = 4 + N(3+3+17) - 12$$

يُنفذ السطر الأول مرة واحدة، في حين يُنفذ السطر الثاني والثالث والرابع N مرة. ولكن في المرة الأخيرة، يكون محتوى السجل CX مساوياً للصفر، ومن ثَم يكون زمن تنفيذ السطر الأخير أقل بـ 12 دوراً. ولذا، ينبغي طرح القيمة 12 من الزمن الكلي. فإذا علمنا أن دور الساعة هو 0.2µs (لأن التردد الخارجي 5MHz)، وأن زمن التأخير المراد تحقيقه هو 5ms (أي 5000µs)، فإننا نستطيع حساب N:

$$5000 = (23N - 8)$$
$$N = \frac{5000-8}{23} = 217 = 0D9h$$

وهكذا نكون قد حصلنا على القيمة الابتدائية N الواجب شحنها في السجل CX لتحقيق حلقة تأخير قدرها 5ms.

ويمكن، إن لم نستطع الحصول على زمن التأخير الكافي من حلقة واحدة، وضع حلقتين متداخلتين Nested Loops كما يلي:

```
        MOV BX, count1
LOOP1:  MOV CX, count2
LOOP2:  LOOP LOOP2
        DEC BX
        JNZ LOOP1
```

فالمبدأ هو تكرار عملية شحن القيمة count2 في السجل CX، ثم إنقاص هذا السجل حتى الصفر عدداً من المرات قدره count1. ويعطى بذلك الزمن الكلي بجداء القيمة count1 بزمن تنفيذ التعليمات الأربع الأخيرة، مضافاً إليه العدد أربعة (وهو زمن السطر الأول). ويمكن لتحديد القيمة الابتدائية اختيار قيمة count2 مساوية FFFFh، وحساب قيمة count1 للحصول على التأخير اللازم.

2-5-3 نقل سلسلة محارف

يمكن أيضاً تنفيذ الحلقة "كرّر-إلى أن" باستخدام تعليمات سلاسل المحارف. فهذه التعليمات مفيدة في برمجيات معالجة النصوص، مثلاً، إذ نحتاج غالباً إلى نقل جملة (مجموعة محارف) من مكان إلى آخر، أو إلى البحث في نص معين عن كلمة ما.

لنفترض أن لدينا سلسلة محارف في مواقع متتالية من الذاكرة، انطلاقاً من الانزياح 2000h في قطاع المعطيات. ولننقل هذه السلسلة إلى المواقع التي تبدأ بالانزياح 2400h.

عندما نبحث عن طريقة تنفيذ هذه العملية بلغة المجمّع، يتبادر إلى الذهن أننا نحتاج إلى مؤشر على السلسلة المصدر لمتابعة المحرف المراد نقله، وإلى مؤشر على السلسلة الوجهة لحفظ موقع كتابة المحرف. ولذلك يُستخدم السجل SI للتأشير على المصدر والسجل DI للتأشير على الوجهة، كما نحتاج إلى عداد للمحارف ونستخدم لذلك السجل CX.

يمكن إذن ترميز هذا البرنامج كما يلي:

كرّر

انقل ثمانيّة من المصدر إلى الوجهة؛

زد مؤشر المصدر SI؛

زد مؤشر الوجهة DI؛

أنقص العداد CX؛

إلى أن يصبح العداد = 0؛

ويكتب البرنامج بلغة المجمّع كما يلي:

```
MOV AX, 0h
MOV DS, AX
MOV ES, AX        ; Initialize ES
MOV SI, 2000h     ; Initialize SI
MOV DI, 2400h     ; Initialize DI
MOV CX, 80h       ; number of bytes in a string
CLD
REP MOVSB
```

تقوم تعليمة MOVSB بنقل محرف من السلسلة المصدر المؤشرعليها بـ SI إلى السلسلة الوجهة المؤشر عليها بـ DI. وتزيد هذه التعليمة آلياً قيمة المؤشرين SI و DI، وهذا ما يفسر وجود التعليمة CLD محو راية الاتجاه. وتدل التعليمة REP على تكرار العملية حتى يصبح محتوى السجل CX معدوماً.

4 الشروط

ثمة بنى شرطية مختلفة Conditions، نعرضها فيما يلي.

1-4 الشرط "إذا كان - افعل"

للبنية الشرطية القاعدية "إذا كان - افعل" If - Then الصيغة التالية:

إذا كان الشرط محققاً افعل

تعليمة 1؛

تعليمة 2؛

...

يمكن تنفيذ هذه البنية بسهولة بواسطة تعليمة القفز المشروط، وفي بعض الحالات لابد من وضع تعليمة للتأثير في الرايات.

مثال:

```
        CMP AX, BX
        JE There
        NOP
        NOP
        NOP
There:  MOV CL, 07h
```

في هذا المثال، تُقارن قيمتا السجلين BX و AX للتأثير في الرايات المشروطة. فإذا رُفعت راية الصفر بعد المقارنة فهذا يدل على أن AX يساوي BX، ويقفز البرنامج إلى السطر ذي اللصاقة There، وإلا فإنه ينفّذ ثلاث تعليمات NOP قبل أن ينتقل إلى السطر There. ولكن في هذا المثال مشكلة، وهي في حال كون عدد التعليمات كبيراً جداً فإن القفز القريب لن يفي بالغرض (وقد نوقشت هذه المشكلة سابقاً).

ولهذا يُفضّل الحل التالي الذي يستخدم تعليمة القفز اللا مشروط JMP، إضافة إلى تعليمة القفز المشروط JNE.

```
        CMP AX, BX
        JNE Fix
        JMP There
Fix:    NOP
        NOP
        NOP
There:  MOV CL, 04h
```

نلاحظ مما سبق أن هذه البنية مشابهة للبنية "مادام – افعل" و "كرر – إلى أن".

2-4 البنية الشرطية "إذا كان – افعل – وإلا"

للبنية الشرطية "إذا كان – افعل – وإلا" If - Then - Else الصيغة التالية:

إذا كان الشرط محققاً افعل

تعليمة 1؛

تعليمة 2؛

...

وإلا

تعليمة 3؛

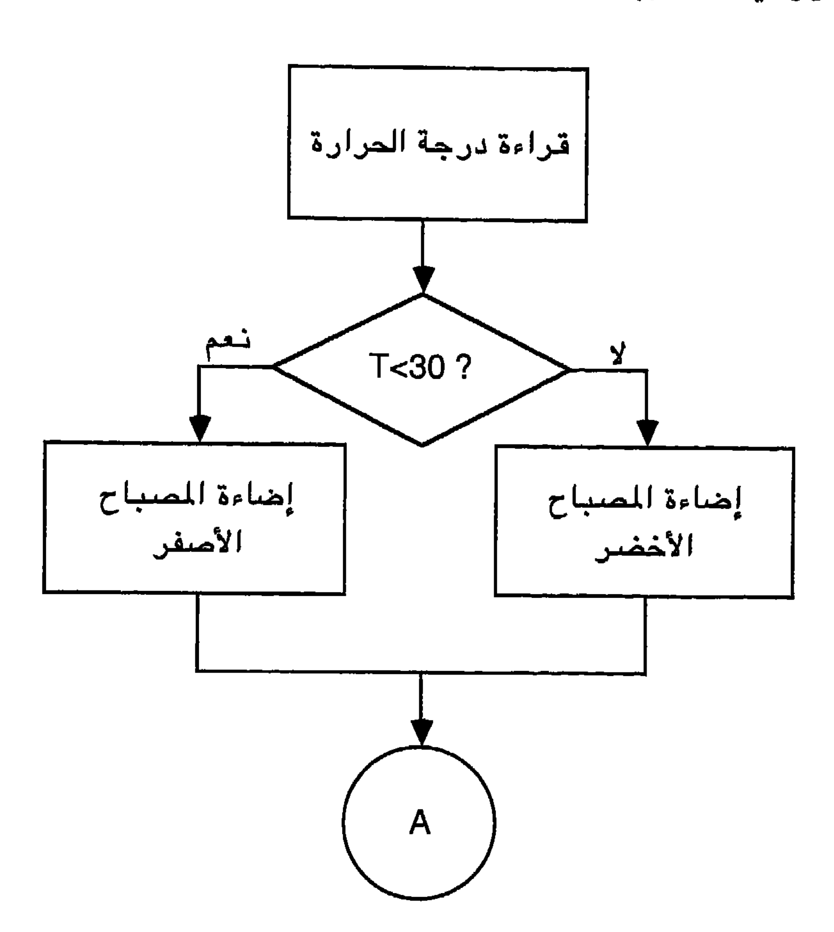
تعليمة 4؛

...

وتُنفذ هذه البنية باستخدام تعليمات القفز المشروط واللامشروط، كما يبين ذلك المثال التالي.

مثال توضيحي:

لنفترض أن لدينا معملاً مؤتمتاً، نريد فيه قراءة محس حرارة و إضاءة المصباح الأخضر إذا كانت درجة الحرارة أكبر من 30°C ، وإضاءة المصباح الأصفر إذا كانت الحرارة دون 30°C . يوضح الشكل 4 الخوارزمية المطلوبة.



الشكل 4: التحكم في درجة حرارة مصنع.

الخوارزمية:

اقرأ المحس؛

إذا كانت درجة الحرارة أكبر من 30 افعل

اضئ المصباح الأخضر؛

وإلا

اضئ المصباح الأصفر؛

البرنامج:

يمكن كتابة البرنامج بلغة المجمّع كما يلي:

```
    MOV DX, FFF8h
    IN AL, DX
    CMP AL, 30
    JB Yellow
    JMP Green
Yellow:
    MOV AL, 01h
    MOV DX, FFFAh
    OUT DX, AL
    JMP Exit
Green:
    MOV AL, 02h
    MOV DX, 0FFFAh
    OUT DX, AL
Exit:
    ...
```

3-4 البنية الشرطية "إذا كان – افعل – وإلا" المتعددة

لهذه البينة الصيغة التالية:

إذا كان الشرط 1 محققاً افعل

تعليمة 1؛

تعليمة 2؛

...

وإلا إذا كان الشرط 2 محققاً افعل

تعليمة 3؛

تعليمة 4؛

...

وإلا

تعليمة 5؛

تعليمة 6؛

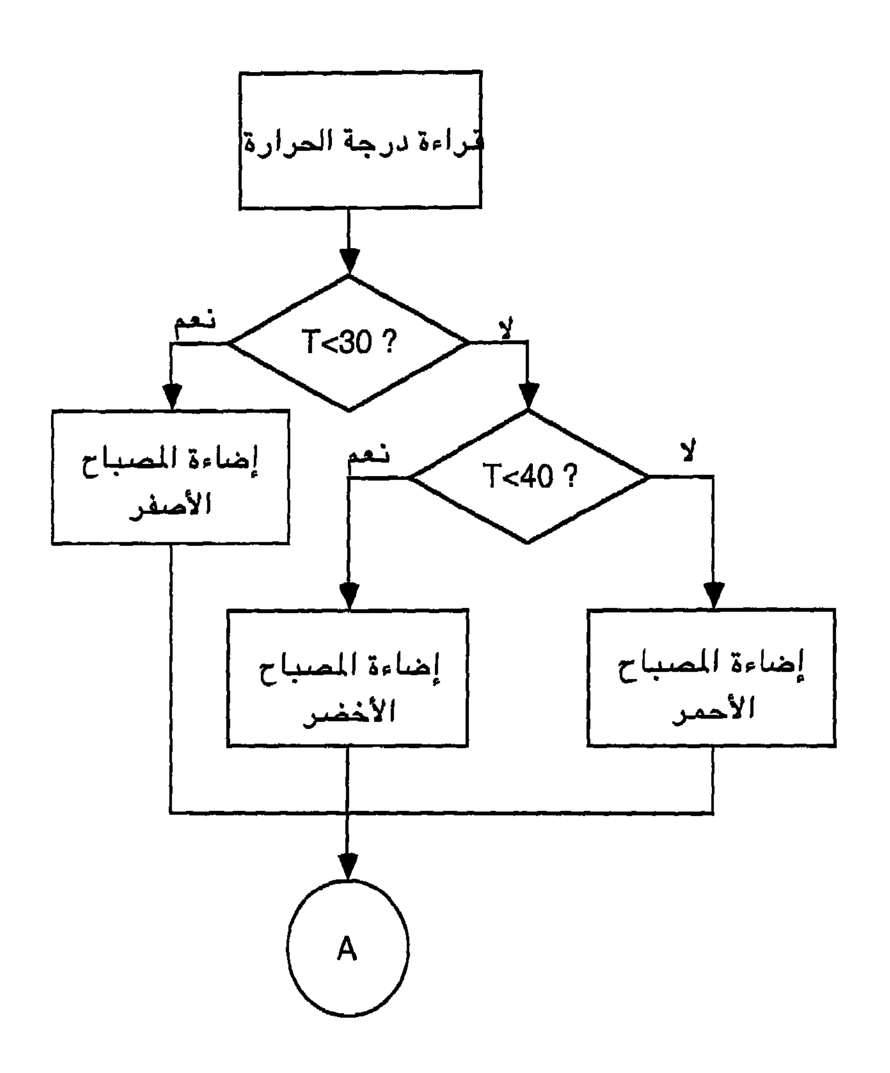
...

تسمح هذه البنية باختيار مهمة من مجموعة مهمات ممكنة، بناء على قيمة بعض المتحولات المقروءة، أو على قيمة مُدخلة من المستثمر مثلاً؛ وتُنفذ باستخدام تعليمات القفز المشروط واللامشروط.

نأخذ لتوضيح ذلك المثال التالي.

مثال:

لنعد ثانيةً إلى المصنع المؤتمت، ولنفترض أن فيه ثلاثة مصابيح: أحمر و أخضر وأصفر. يُضيء المصباح الأحمر إذا كانت درجة الحرارة أكبر من 40°C، أما المصباح الأخضر فيضيء إذا كانت درجة الحرارة بين 30°C و 40°C، ويُضيء المصباح الأصفر عندما تقل درجة الحرارة عن 30°C. تُعطى الخوارزمية بالشكل 5.



الشكل 5: التحكم في درجة حرارة مصنع ذي ثلاثة مصابيح.

أما البرنامج بلغة المجمّع فهو:

```
    MOV DX, FFF8h    ; point DX at input port
    IN AL, DX        ; read temperature
    MOV DX, FFFAh    ; point DX to output port
    CMP AL, 1Eh
    JB Yellow
    CMP AL, 28h
    JB Green
Red:
    MOV AL, 04h      ; temperature ≥ 40
    OUT DX, AL       ; load code to light red lamp
    JMP Exit
Yellow:
    MOV AL, 01h      ; temperature ≤ 30
    OUT DX, AL
    JMP Exit
Green:
    MOV AL, 02h      ; temperature ≥ 30 and temperature ≤ 40
    OUT DX, AL
Exit:
    ...
```

نقرأ في هذا البرنامج قيمة المحس ثم نقارنها بالقيمة 30°C، فإذا كانت أصغر منها وجبت إضاءة المصباح الأصفر، وإلا فإننا نقارن ثانيةً بالقيمة 40°C، فإذا كانت أصغر منها وجبت إضاءة المصباح الأخضر، وإلا فيجب إضاءة المصباح الأحمر.

نفترض أن المصابيح موصولة إلى المخارج كما يلي:

- اللون الأحمر موصول إلى الخانة 2؛
- اللون الأخضر موصول إلى الخانة 1؛
- اللون الأصفر موصول إلى الخانة 0.

نذكر أخيراً أنه يمكن بالطريقة ذاتها تنفيذ بنية الانتقاء المتعدد CASE، باستخدام تعليمات القفز المشروط واللامشروط.

5 التعليمات الموسّعة

توجد طريقتان لمنع تكرار مجموعة من التعليمات عدداً كبيراً من المرات في البرنامج ذاته. تنص الطريقة الأولى على جمع هذه التعليمات في إجرائية Procedure (أو برنامج فرعي) مستقل (انظر الفقرة التالية)، ومن ثم طلب هذا البرنامج الفرعي بواسطة التعليمة CALL عند الحاجة إليه في البرنامج الرئيسي. إن الميزة الرئيسية لاستخدام الإجرائية، كما سنرى، هي أن الرموز الاثنانية المقابلة لتعليمات الإجرائية ستُخزن في مكان واحد في الذاكرة. ولكن مثالب هذه الطريقة تكمن في الحاجة إلى المكدس، وفي الزمن الإضافي اللازم عند طلب الإجرائية وعند العودة منها، وهذا ما يؤدي إلى تقليص سرعة التنفيذ.

أما الطريقة الثانية المعتمدة للحيلولة دون تكرار التعليمات، فهي وضع تلك التعليمات في وحدات برمجية تسمى التعليمات الموسّعة Macro. وهذه الطريقة مفيدة عندما يكون عدد التعليمات المكررة قليلاً، أو عندما لا يمكن جمعها في إجرائية ذات وظيفة محددة تماماً. فالتعليمة الموسّعة إذن هي مجموعة من تعليمات المعالج التي تُعطى اسماً في بداية البرنامج. ويمكن باستخدامها جعل البرنامج سهل القراءة.

وفي كل مرة تُطلب بها تعليمة موسّعة خلال البرنامج الرئيسي، يقوم المجمع بحشر مجموعة التعلميات المكوِّنة لها بدلاً من تعليمة الطلب. ومن ثم، يولّد المجمِّع رموزاً ثنائية مقابلة لتعليمات التعليمة الموسعة في كل مكان تُطلب فيه. وهذا ما يُعرف بنشرالتعليمات الموسّعة Expanding. ومن جهة أخرى، لا تُعد طريقة التعليمات الموسّعة طريقة اقتصادية في حجم الذاكرة. ولكن، لما كان المجمِّع يستبدل التعليمات المكوّنة للتعليمة الموسعة بتعليمة الطلب، فإن تنفيذها لا يحتاج إلى زمن إضافي عند الطلب أو العودة.

تختلف طريقة تعريف التعليمات الموسعة تبعاً لنوع المجمّع المعتمد. وسنذكر هنا مثالاً يعتمد صيغة المجمّع MASM المصنّع من شركة MICROSOFT.

مثال:

نود كتابة تعليمة موسعة تحفظ رايات المعالج وسجلاته في المكدس، وكتابة تعليمة موسعة أخرى تقوم باسترجاع هذه السجلات والرايات وفق الترتيب المناسب.

```
PUSH_ALL MACRO
    PUSHF
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH BP
    PUSH SI
    PUSH DI
    PUSH DS
    PUSH ES
    PUSH SS
ENDM
```

تعرّف العبارة PUSH_ALL MACRO تعليمة موسّعة اسمها PUSH_ALL، وتدل على بداية تعريف التعليمة الموسّعة، في حين تدل العبارة ENDM على نهاية التعليمة الموسّعة.ويمكن الآن استخدام هذه التعليمة الموسّعة في البرنامج كما يلي:

```
    MOV BX, 20h
    PUSH_ALL
    MOV AX, 10h
```

عندما يصل المجمّع إلى سطر التعليمة الموسّعة، يستعيض عنها بالرموز الثنائية المقابلة للتعليمات المكوّنة لها.

يمكن بلغة المجمّع كتابة تعليمات موسّعة بحيث تقبل بعض المعاملات. وتُعيّن هذه المعاملات عند تعريف التعليمة الموسعة بأسماء رمزية. تُستخدم أسماء هذه المعاملات في التعليمات المكونة للتعليمة الموسّعة. أما عند طلب التعليمة الموسّعة، فيُستعاض عن أسماء المعاملات الرمزية بالقيم الحقيقية المراد تمريرها. فتُنفذ التعليمات المكوِّنة للتعليمة الموسّعة بالاعتماد على تلك القيم.

لنأخذ المثال التالي على سبيل الإيضاح. نفترض أننا نريد كتابة تعليمة موسّعة تنقل مجموعة من المحارف ASCII من مكان في الذاكرة إلى مكان آخر. ولإجراء ذلك، تُستعمل تعليمة MOVS التي تتطلب وضع المؤشر SI على السلسلة المصدر، والمؤشر DI على السلسلة الوجهة، ثم شحن السجل CX بعدد المحارف الواجب نقلها. تُكتب إذن التعليمة الموسعة على النحو التالي:

```
MOVE_ASCII   MACRO    NUMBER, SOURCE, DESTINATION
     MOV CX, NUMBER          ; number of characters
     LEA SI, SOURCE
     LEA DI, DESTINATION
     REP MOVSB
 ENDM
```

تمثل الأسماء الرمزية NUMBER, SOURCE, DESTINATION معاملات التعليمة الموسعة. و عند طلب التعليمة المؤسعة في البرنامج، يُستعاض عن المعامِلات بالقيم الفعلية. فمثلاً، إذا طلبت التعليمة على النحو الآتي:

MOVE_ASCII 03Dh, BlockStart, BlockEnd

فإن المتحول NUMBER يأخذ القيمة 3Dh، وتصبح قيمة المتحول SOURCE العنوان BlockStart، وكذلك يأخذ المتحول DESTINATION قيمة العنوان BlockEnd .

6 البرامج الفرعية والإجرائيات

عندما يكثر ورود مجموعة من التعليمات في برنامج ما يصبح من الأفضل جمعها في وحدة واحدة تُسمى برنامجاً فرعياً Subprogram.

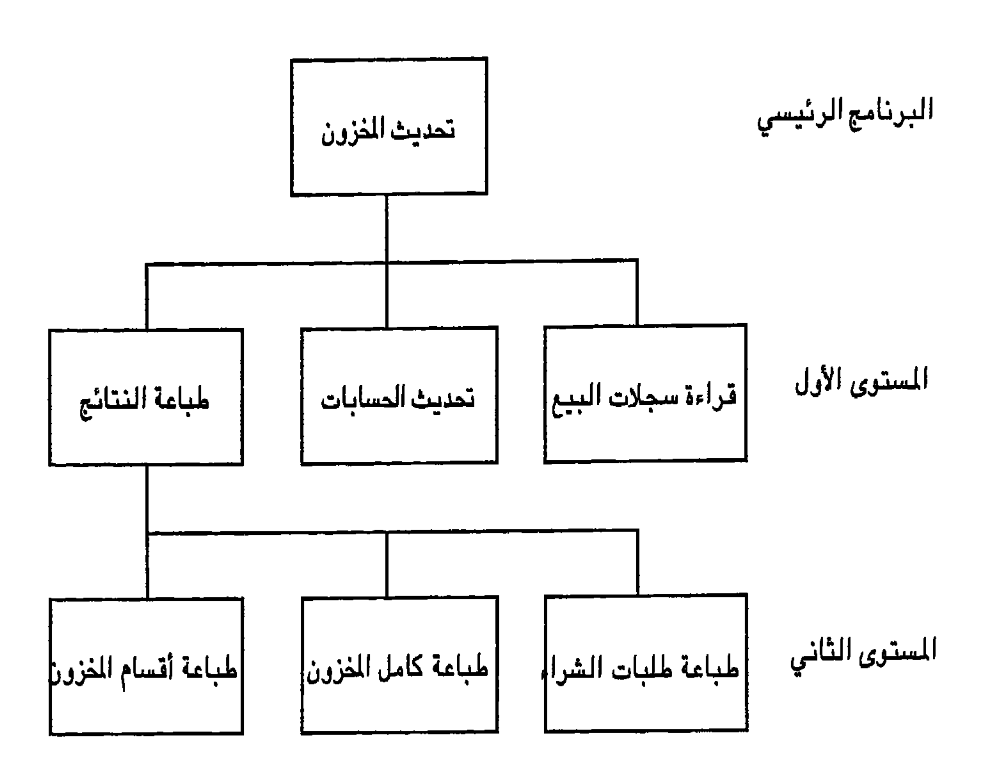
توفّرالبرامج الفرعية للمبرمج عدة مِيْزات منها:

- الكتابة المضغوطة للبرنامج: أي تقليص حجم الملف المصدري، إذ يُستغنى عن مجموعة التعليمات المتكررة بتعليمة واحدة وهي طلب للبرنامج الفرعي.
- تنظيم البرنامج: يغدو البرنامج المكتوب أسهل قراءةً وفهماً، وذلك لأن لكل برنامج فرعي وظيفة محددة تجعل متابعة مراحل البرنامج أمراً يسيراً.
- سهولة التطوير: يمكن عند كتابة البرنامج اختبار كل برنامج فرعي على حدة، والتوثق من أدائه، قبل مكاملته في البرنامج النهائي.

يُطلب البرنامج الفرعي بتعليمة CALL لتنفيذ مجموعة التعليمات التي يتضمنها. وتسمى البرامج الفرعية، طبقاً لمصطلحات شركة INTEL، إجرائيات Procedures.

وهناك مِيْزة أخرى لاستخدام الإجرائيات، وهي تسهيل اعتماد النهج التنازلي في حل مسائل البرمجة. ففي هذا النهج، تُعرّف المسألة جيداً، ثم يُجزأ العمل الكلي في واحدات أصغر، وتُقسم بدورها إلى أجزاء أصغر فأصغر، إلى أن تصبح الخوارزمية جلية تماماً. يظهر الشكل 6 مثالاً على التمثيل التراتبي Hierarchical Representation.

يكمن المبدأ الرئيسي في تقسيم المسألة الضخمة إلى أجزاء سهلة التداول، تُكتب مستقلةً بعضها عن بعض وتختبر وحدها. ثم تُجمع هذه الأجزاء المتناثرة في البرنامج الرئيسي، الذي ينحصر دوره عندئذٍ في طلب هذه الإجرائيات حين الحاجة إليها.



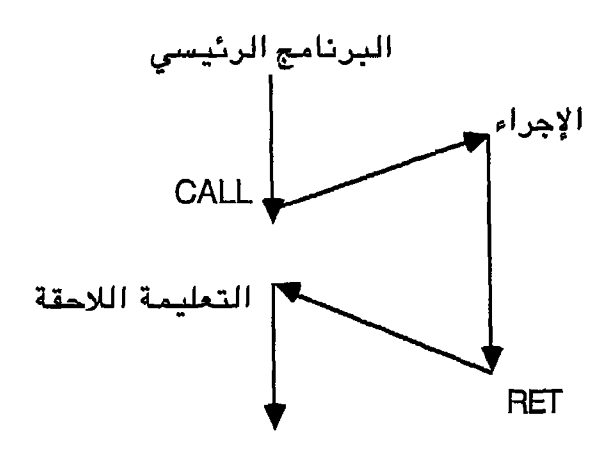
الشكل 6: مثال على التمثيل التراتبي.

ويضيف هذا النهج ميزة أخرى، وهي أن الشخص الذي يقرأ البرنامج يحصل على فكرة عامة عن مهمات البرنامج، ثم يستطيع بعدها الغوص في تفاصيل كل برنامج جزئي على حدة.

1-6 طريقة عمل البرامج الفرعية

تعمل الإجرائيات على النحو التالي: عندما يصادف المعالج تعليمة CALL في البرنامج الرئيسي، فإنه يشحن عنوان بداية الإجرائية في مؤشر التعليمات IP. وعندئذٍ تكون التعليمة المنفَذة هي أول تعليمة في الإجرائية. يستمر تنفيذ التعليمات إلى أن يصادف المعالج تعليمة RET التي تشير إلى نهاية الإجرائية، فتعود

عندها السيطرة إلى البرنامج الرئيسي. فينتقل المعالج إلى التعليمة التالية لتعليمة طلب الإجرائية CALL. وكما يظهر في الشكل 7، يمكن لإجرائية ما أن تطلب إجرائية أخرى وهذا ما يُسمى الإجرائيات المتداخلة Nesting Procedures. وفي هذه الحالة، فإن تعليمة RET المصادفة في الإجرائية ذات المستوى الأدنى ستعيد السيطرة للإجرائية التي تعلوها مباشرةً، وهكذا...



الشكل 7: تنفيذ الإجرائيات في البرنامج.

والسؤال هنا كيف يستطيع المعالج أن يعرف المكان الذي سيعود إليه بعد تعليمة RET؟ في الواقع، عند ما ينفذ المعالج تعليمة CALL فإنه يخزن عنوان التعليمة التالية في المكدس، وعندما ينفذ تعليمة RET في نهاية الإجرائية فإنه يستعيد ذلك العنوان من المكدس ويشحنه في مؤشر التعليمات IP.

1-1-6 تعليمة طلب برنامج فرعي

لهذه التعليمة عملان: فهي تخزّن أولاً عنوان التعليمة التي تلي تعليمة Call في المكدس، وهو ما يسمى عنوان العودة، لأنه سيُشحن ثانيةً في مؤشر التعليمات عند الانتهاء من تنفيذ الإجرائية. وثانياً،

تغيّر محتوى مؤشر التعليمات للقفز إلى عنوان الإجرائية المطلوب. وفي بعض الأحيان، تخزّن تعليمة CALL في المكدس الانزياح ومحتوى سجل القطاع عند القفز خارج القطاع الحالي. وفي تلك الحالة، فهي تشحن مؤشر التعليمات وقطاع التعليمات بالقيم المناسبة.

يمكن أن نميّز بين أربعة أنواع لتعليمة CALL، وذلك تبعاً لطريقة الحصول على عنوان بداية الإجرائية:

- تعليمة CALL القريبة المباشرة:

نحصل على عنوان بداية الإجرائية بجمع الانزياح -ذي الإشارة- المحدد بالتعليمة (على 16 خانة) إلى مؤشر التعليمات IP. يمكن بهذه التعليمة طلب إجرائية تبعد مسافة تقع بين 768 32- و 767 32+ ثمانيّة عن الموقع الحالي.

- تعليمة CALL القريبة غير المباشرة:

يبقى المعالج في هذه الحالة أيضاً ضمن القطاع ذاته. ولكنه يحصل على قيمة مؤشر التعليمات IP من سجل أو من موقعين متتاليين في الذاكرة.

- تعليمة CALL البعيدة المباشرة:

يخرج المعالج بهذه التعليمة إلى قطاع آخر بحثاً عن بداية الإجرائية. ولذا ينبغي تغيير مؤشر التعليمات IP وسجل قطاع البرنامج CS معاً. ويحصل المعالج على القيم الجديدة من التعليمة التي ستُرمز على 5 ثمانيّات. فتدل الثمانيّة الثانية والثالثة على مؤشر التعليمات، وتحتوي الثمانيّة الرابعة والخامسة على قيمة سجل قطاع البرنامج الجديدة.

- تعليمة CALL البعيدة غير المباشرة:

في هذا النمط يغيّر المعالج القطاع الحالي. ولكنه يحصل على القيمة الجديدة من الذاكرة. فيدل أول موقعين على مؤشر التعليمات IP، في حين يحتوي الموقعان التاليان على قيمة سجل قطاع البرنامج الجديدة.

2-1-6 تعليمة العودة من برنامج فرعي

كما ذكرنا سابقاً يعود المعالج من برنامج فرعي (أو إجرائية) بالتعليمة RET. تسترجع هذه التعليمة القيم المخزّنة في المكدس وتشحنها في مؤشر التعليمات IP وفي قطاع البرنامج CS. فإذا استخدمت تعليمة CALL من النوع القريب (داخل القطاع) فإن تعليمة RET تسترجع كلمة واحدة من المكدس (مرمزة على 16 خانة)، وتشحنها في مؤشر التعليمات IP. أما إذا كانت من النوع البعيد، فإن تعليمة RET تسترجع كلمتين من المكدس، إذ تشحن الكلمة الأولى في مؤشر التعليمات IP والثانية في قطاع البرنامج CS.

توجد إذن أربعة أنواع من تعليمة RET:

- تعليمة RET القريبة:

وهي تنسخ كلمة واحدة من المكدس (ذات 16 خانة) إلى مؤشر التعليمات IP.

- تعليمة RET القريبة مع انزياح:

وهي مثل سابقتها، ولكنها تجمع إلى مؤشر التعليمات قيمة ثابتة محددة بالتعليمة.

- تعليمة RET البعيدة:

تنسخ كلمتين من المكدس إلى مؤشر التعليمات وسجل قطاع البرنامج.

- تعليمة RET البعيدة مع انزياح:

وهي مثل سابقتها ولكنها تضيف إلى مؤشر التعليمات قيمة ثابتة محددة بالتعليمة.

2-6 استخدام المكدس في البرامج الفرعية

ذكرنا أن المكدس جزء من الذاكرة يُستخدم لتخزين عناوين العودة. كما يمكن الاستفادة منه، أثناء تنفيذ إجرائية ما، في تخزين محتوى السجلات المستخدمة في البرنامج الرئيسي. أما العمل

الثالث للمكدس فهو حفظ المعطيات أو العناوين المعالَجة داخل إجرائية ما.

يدير المعالج 8086 المكدس بواسطة سجلين: سجل قطاع المكدس SS وسجل مؤشر المكدس SP. يُستخدم هذان السجلان لتوليد العناوين الحقيقية لمواقع موجودة ضمن المكدس. فالسجل SS يحوي الأوزان العليا لعنوان الموقع، في حين يحوي مؤشر المكدس انزياح الموقع بالنسبة إلى بداية قطاع المكدس. وفي كل الأحوال، فإن حجم القطاع لا يمكن أن يتجاوز 64KBytes. ولذا، ينبغي عند استخدام المكدس في برنامج ما وضع قيمة ابتدائية مناسبة في هذين السجلين في بداية البرنامج.

عند التخزين في المكدس، نبدأ دوماً بالموقع ذي العنوان الأعلى، ثم ننزل متجهين نحو الموقع ذي العنوان الأدنى. وهذا يعاكس طريقة التخزين في الذاكرة، والتي نبدأ فيها بالعنوان الأدنى متجهين نحو الأعلى. ولذلك، نحتاج دوماً إلى مؤشر نحو قمة المكدس نسميه Stack Top. يظهر الشكل 8 مثالاً على تنظيم المكدس في الذاكرة.

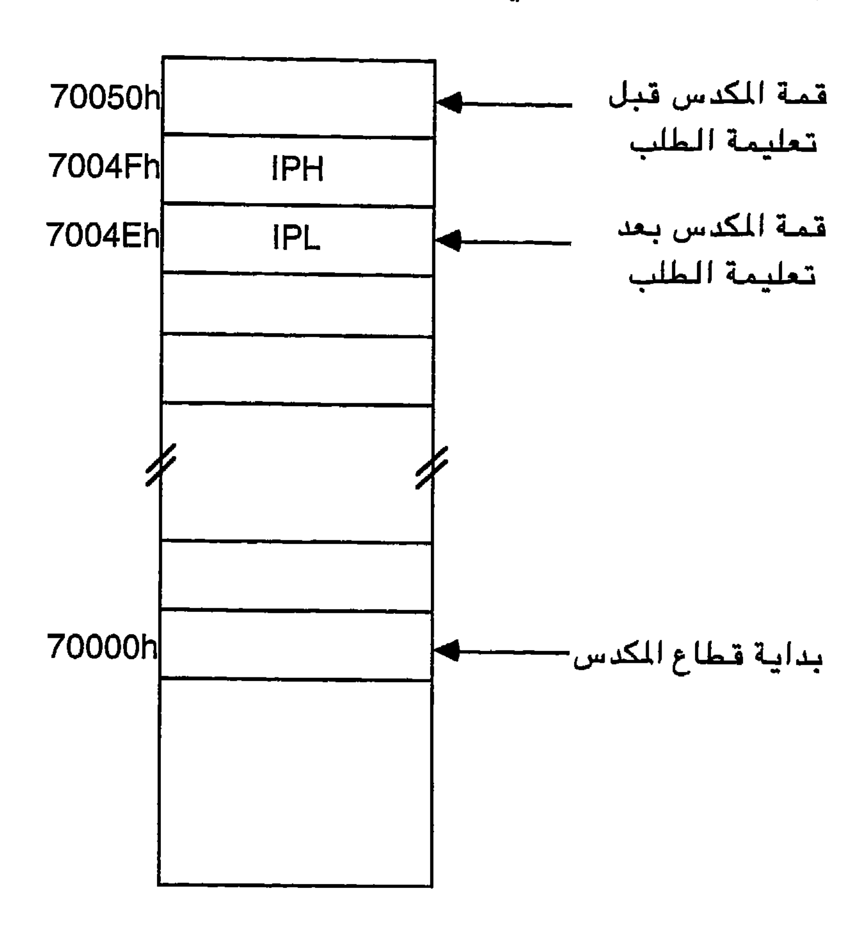
في هذا المثال، اختير العنوان 70000h اعتباطاً ليكون بداية قطاع المكدس. واختير طول المكدس 40 كلمة (أي 80 ثمانيّة)، وأُلصقت اللصاقة StackTop بالعنوان 70050h، وهو العنوان الذي يلي نهاية المكدس. يوضّح مقطع البرنامج التالي طريقة استهلال سجلي المكدس.

```
MOV AX, StackHere
MOV SS, AX
LEA SP, StackTop
```

حيث StackHere و StackTop لصاقتان تشيران إلى سجل قطاع المكدس SS، وانزياح بداية المكدس SP على الترتيب.

ينبغي أن يُشحن قطاع المكدس خلال البرنامج، كما فعلنا سابقاً فيما يخص قطاع البرنامج والقطاع الإضافي، ولا يمكن إجراء الشحن مباشرةً، بل يجب المرور بأحد سجلات المعالج. ولذا، فقد شُحن

العنوان أولاً في السجل AX ثم نُقل إلى قطاع المكدس SS. كما شُحن انزياح اللصاقة StackTop في مؤشر المكدس بغرض الاستهلال.



الشكل 8: تنظيم المكدس في الذاكرة.

3-6 الطلب القريب للإجرائيات

سنعرض هنا مثالاً يوضح طريقة طلب إجرائية طلباً قريباً.

1 *تعريف المسألة وكتابة الخوارزمية:*

يمكن كتابة حلقات التأخير -كالمثال الذي عُرض سابقاً- في صيغة إجرائيات تُطلب من أي موقع في البرنامج. فلنكتب على سبيل المثال برنامجاً يقرأ 100 كلمة من معبر بفاصل زمني قدره 1ms بين الكلمة والأخرى، ثم يخزّن الأوزان الدنيا فقط من هذه القيم في جدول في الذاكرة:

كرّر

احصل على عينة من المعبر؛

اعزل الأوزان الدنيا؛

خزّن في الجدول؛

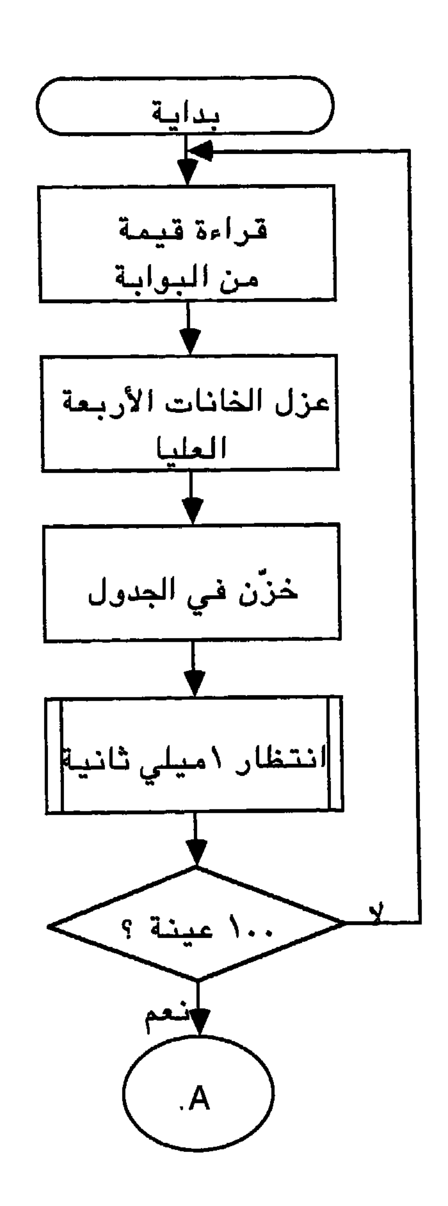
انتظر 1ms؛

إلى أن يصبح عدد العينات = 100؛

يظهر الشكل 9 المخطط التدفقي المعتمد.

تتضح إذن ضرورة وضع مؤشر على جدول التخزين، ووضع عداد للعينات المقروءة تُكتب فيهما القيم الابتدائية في بداية البرنامج. نزيد، بعد قراءة كل عينة وتخزينها في الذاكرة، مؤشر الجدول بمقدار واحد وننقص العداد بمقدار واحد أيضاً. ثم تُطلب الإجرائية WAIT لانتظار 1ms.

تُكتب خوارزمية الانتظار مستقلةً عن البرنامج الرئيسي، وهذا ما يجعل تسلسل البرنامج أكثر وضوحاً؛ إذ لاداعي للغوص في التفاصيل عند قراءة البرنامج الرئيسي. وبعد العودة من الإجرائية، يُختبر شرط التوقف ويقفز المعالج إلى بداية البرنامج في حال عدم تحققه.



الشكل 9: المخطط التدفقي لبرنامج القراءة من معبر مع الانتظار.

2 كتابة البرنامج:

نعرض فيما يلي البرنامج الكامل الموافق لهذا المثال. وهو كما نلاحظ أطول من بقية الأمثلة، ومع ذلك، على القارئ أن يحاول فك رموز هذا البرنامج للتدرب على تحليل برامج الغير.

```
PRESSURE_PORT EQU 0FFF8H

DATA_HERE SEGMENT
PRESSURES DW 100 DUP(0)          ; set up array of 100 words
DATA_HERE ENDS

STACK_HERE SEGMENT STACK
DW 40 DUP(0)                     ; set stack length of 40 words
STACK_TOP LABEL WORD
STACK_HERE ENDS

CODE_HERE SEGMENT
ASSUME CS:CODE_HERE, DS:DATA_HERE, SS:STACK_HERE
START:
    MOV AX, DATA_HERE            ; initialize data segment register
    MOV DS, AX
    MOV AX, STACK_HERE           ; initialize stack segment register
    MOV SS, AX;
    MOV SP, OFFSET STACK_TOP     ; initialize stack pointer to stacktop
    LEA SI, PRESSURES            ; point SI to start of array
    MOV BX, 100                  ; load BX with number of samples
    MOV DX, PRESSURE_PORT        ; point DX at input port
NEXT_VALUE:
    IN AX, DX                    ; read data from port
    AND AX, OFFFh                ; mask upper 4 bits
    MOV [SI], AX                 ; store data word in array
    CALL WAIT_1MS                ; delay of 1 ms
    INC SI                       ; point SI at next location in array
    INC SI
    DEC BX                       ; decrement sample counter
    JNZ NEXT_VALUE               ; repeat until 100 samples done
STOP:
    NOP
```

```
WAIT_1MS: PROC NEAR
    MOV CX, 23F2h          ; load delay constant into CX
HERE:
    LOOP HERE              ; load until CX=0
    RET
WAIT_1MS: ENDP
CODE_HERE: ENDS

END
```

نصرّح في البداية بوجود قطاع المعطيات بواسطة العبارتين: DataHere Segment و DataHere ENDS. أما العبارة PRESSURES DW 100 DUP(0) فهي تحجز 100 كلمة في الذاكرة لتخزين العينات المقروءة من محس الضغط، وتعطي هذه المواقع قيمة ابتدائية تساوي الصفر. إن ضم هذه المواقع إلى قائمة الاستهلال أمر غير إلزامي، لأن المعالج سيكتب فيها قيماً جديدة. إلا أن وجود الأصفار قد يساعد في تنقيح البرنامج.

نصرّح بوجود قطاع المكدس بواسطة العبارتين: StackHere Segment و StackHere ENDS. وتحجز العبارة DW 40 DUP(0) أربعين كلمة في هذا القطاع من الذاكرة وتعطيها قيمة ابتدائية تساوي الصفر. وليس من الضروري في هذه الحالة أيضاً ضم مواقع المكدس إلى قائمة الاستهلال لأن المعالج سيكتب فيها أثناء البرنامج. تُرفِق العبارة StackTop LABEL WORD اسماً بالموقع ذي العنوان الزوجي التالي لأعلى عنوان في قطاع المكدس.

والآن لننظر في البرنامج الرئيسي. ينبغي لنا أن نُعلم المجمِّع عن القطاعات المنطقية المستخدمة في البرنامج، ولهذا تكتب العبارة التالية:

ASSUME CS: CodeHere, DS: DataHere, SS:StackHere

إن عبارة ASSUME لا تضع قيمة ابتدائية في السجلات DS, SS, CS ولكنها توجه عمل المجمِّع فقط.

ينبغي لنا، لاستهلال سجلات القطاعات، أن ننقل القيمة أولاً إلى أحد سجلات المعالج، ثم نشحنها في سجل القطاع الموافق. بعد ذلك، تبدأ المعالجة الفعلية للمسألة المطروحة. فتُشحن قيمة ابتدائية في المؤشر SI بالتعليمة LEA SI, PRESSURES، ليؤشر على الموقع الأول في الجدول PRESSURES. ونختار السجل BX، مثلاً، ليكون عداداً للعينات المتبقية. ويُشحن ذلك السجل بالقيمة الابتدائية 100.

نقرأ عينة من المعبر FFF8h ونعزل الخانات الدنيا بالعملية المنطقية AND مع القيمة الثابتة 00FFh. تحتوي النتيجة على الخانات الدنيا فقط، أما الخانات العليا فتكون معدومة. وسبب ذلك هو أن القيم المقروءة من المحس تُرمز على 8 خانات فقط، ولذا يمكن إعطاء الخانات العليا القيمة صفر من غير فقد المعلومات. تحول هذه العملية دون قراءة قيم عشوائية في الخانات العليا والتي قد تنتج مثلاً عن الضجيج.

تُشحن هذه القيمة في الجدول باستخدام المؤشر SI. ويُطلب بعد ذلك البرنامج الفرعي WAIT بواسطة تعليمة طلب قريبة مباشرة، لأن الإجرائية تقع في القطاع ذاته.

نستعمل لكتابة البرنامج الفرعي WAIT التوجيهين PROC و ENDP ليحيطا بتعليمات الاجراء. ويُطلب البرنامج الفرعي، المكتوب على هذا النحو، بالاسم المذكور في سطر التوجيه PROC.

نحصل على التأخير 1ms بشحن السجل CX بالقيمة 23F2h، وتُجرى حلقة لإنقاص هذا السجل حتى الصفر. ومن الجدير بالذكر أن الإجرائية تنتهي دوماً بتعليمة RET التي تعيد السيطرة إلى البرنامج الطالب.

بعد العودة إلى البرنامج الرئيسي، نزيد المؤشر SI مرتين ليؤشر على الكلمة التالية وننقص المؤشر BX بمقدار واحد. ثم تُفحص راية الصفر، فإن لم تكن مرفوعة قفز المعالج إلى اللصاقة NextValue. ولما كانت تعليمة CALL تنسخ القيمة اللازمة لمؤشر التعليمات IP إلى المكدس، فإن تعليمة RET تنسخ هذه القيمة من المكدس نحو IP.

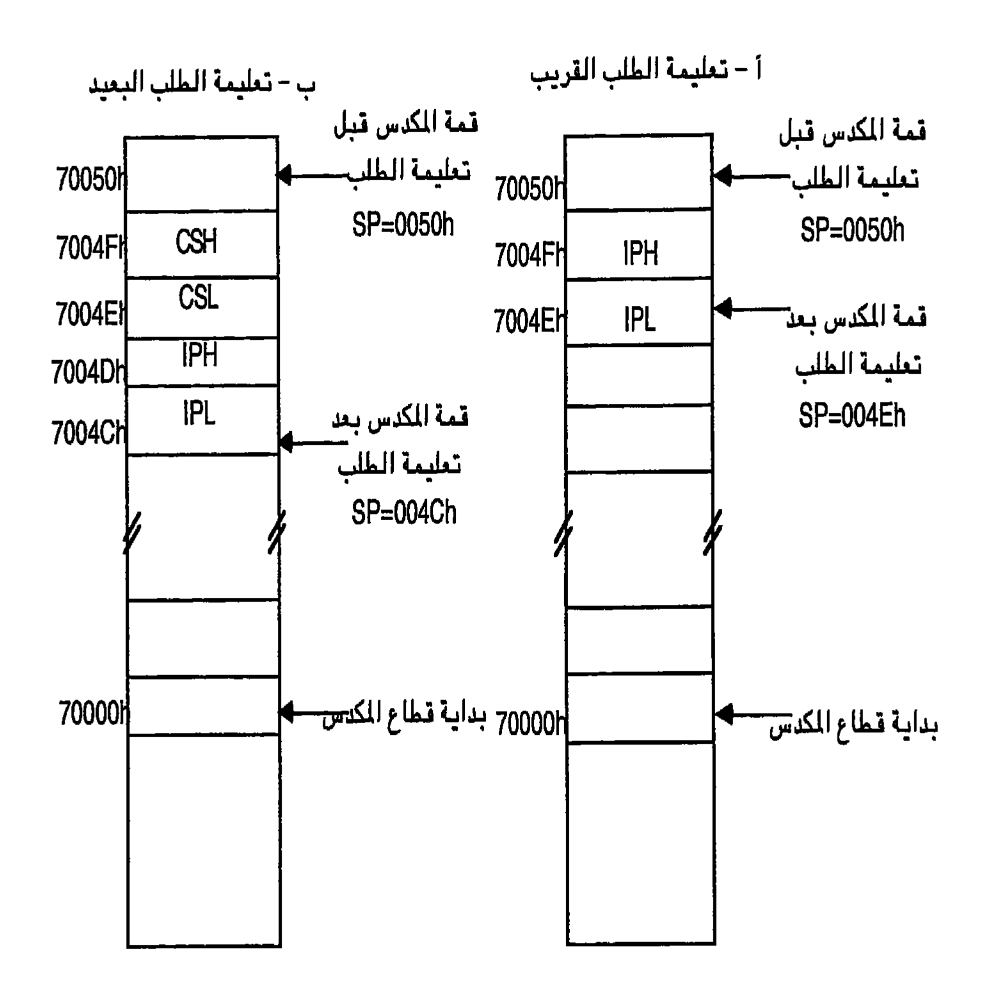
4-6 عمل المكدس

لإيضاح عمل المكدس أثناء تعليمتي CALL و RET، سنفترض أن بداية المكدس هي 70000h، فيكون إذن محتوى سجل قطاع المكدس 7000h. وسنفترض أن المكدس يحتوي على 80 ثمانيّة (أي إنه يمتد حتى العنوان 7004Fh).

عند كتابة الكلمات في المكدس، نضع الكلمة الأولى في العنوان الأعلى. ولذا، فأول كلمة تُكتب في العنوانين 7004Eh و 7004Fh، أي إن المكدس يُملأ من الأعلى نحو الأسفل. ونستخدم مؤشر المكدس SP للإشارة إلى قمة المكدس (وهي موقع آخر كلمة مكتوبة). بعد تنفيذ تعليمة CALL، يتقدم مؤشر التعليمات IP آلياً ليشير إلى التعليمة التالية. فعندما تكون تعليمة CALL من النوع القريب، فإن المعالج ينسخ أولاً محتوى السجل IP إلى المكدس، ثم ينقص ثانيةً مؤشر المكدس بـ2 (انظر الشكل 10).

يمثل محتوى السجل IP عنوان العودة، فإذا كان مؤشر المكدس يحوي القيمة 0050h قبل تعليمة CALL، فإنه يحتوي بعد تنفيذها على القيمة 004Eh. وتُنسخ الثمانيّة الدنيا إلى العنوان الأدنى (أي 004Eh)، في حين تُنسخ الثمانيّة العليا إلى الموقع الذي يليه، أي 004Fh. وبالعكس، عندما تُنفذ تعليمة RET في نهاية الإجرائية يشحن المعالج عنوان العودة من قمة المكدس إلى مؤشر التعليمات IP، ويزيد مؤشر المكدس بمقدار 2، فيعود محتواه إلى القيمة 0050h من جديد.

وتجدر الملاحظة أن تنفيذ تعليمة CALL من النوع البعيد يؤدي إلى إنقاص مؤشر المكدس بمقدار 4 لكي يتسع لكلمتين هما: السجل IP والسجل CS. ثم يعود هذا المؤشر إلى قيمته الأولى عندما تُنفذ تعليمة RET.



الشكل 10: أثر التعليمتين CALL و RET على المكدس.

5-6 استخدام تعليمتي الدفع داخل/خارج المكدس

يمكننا استخدام المكدس لتخزين محتويات السجلات المستخدمة في الإجرائيات. ففي مثالنا السابق استخدمنا السجل BX كعداد للعيّنات المتبقية، إلا أن من الأفضل استخدام السجل CX لهذا الغرض، لكي نستعيض بالتعليمة LOOP عن التعليمتين JNZ و DEC.

ولم نستطع فعل ذلك لأن السجل CX مستخدم في الإجرائية WAIT: فعند تخزين أي قيمة في هذا السجل، فإنها تُمحى في الإجرائية. ولذا، نحتاج إلى طريقة تسمح باستخدام السجلات ذاتها في البرنامج الرئيسي والإجرائية معاً. و هذا هو دور التعليمتين PUSH وPOP.

تنقص PUSH مؤشر المكدس بمقدار 2، وتنسخ محتويات السجل أو موقع الذاكرة المحدد بالتعليمة إلى المكدس. فمثلاً، تؤدي التعليمة PUSH CX إلى إنقاص مؤشر المكدس بمقدار 2 ونسخ محتويات السجل CX إلى المكدس. والسؤال كيف يمكن استرجاع القيمة المخزّنة وشحنها في CX ثانيةً. نستطيع ذلك بواسطة التعليمة POP التي تنسخ كلمة من قمة المكدس وتخزّنها في السجل أو الموقع المحدد بالتعليمة، وتزيد مؤشر المكدس بمقدار 2. فمثلاً تعيد التعليمة POP CX القيمة المخزنة في المكدس إلى CX وتزيد مؤشر المكدس بمقدار 2.

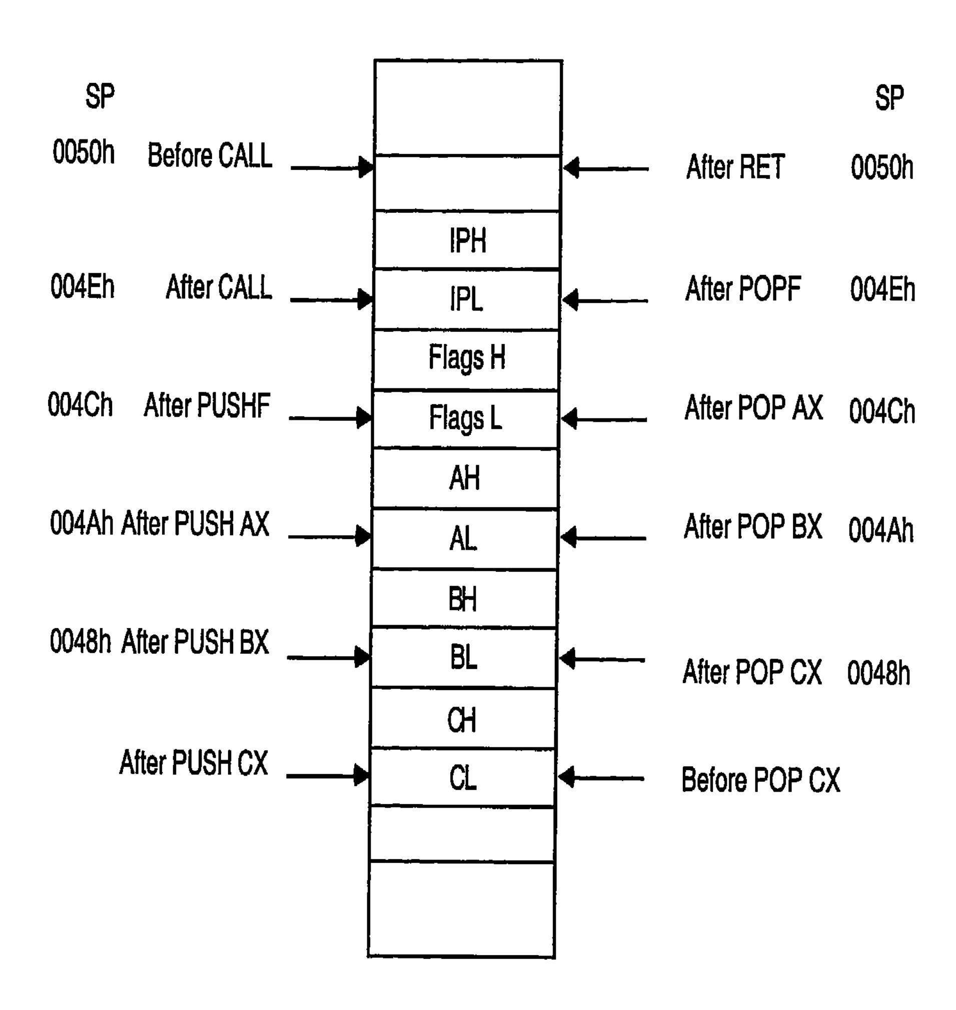
وفي الواقع، يمكن دفع أي سجل ذي 16 خانة إلى المكدس (AX, BX, CX, DX, DI, SI, SP, BP, CS, SS, ES, DS)، أو أي موقعين متتاليين في الذاكرة. وكذلك تُسترجع أي قيمة من المكدس وتُخزّن في أحد السجلات المرمزة على 16 خانة أو في موقعين متتاليين في الذاكرة.

يتبع المكدس في عمله آلية تُسمى «الداخل أخيراً يخرج أولاً» LIFO (Last In First Out)، وهو نظام إدارة شائع. فمثلاً تُرتب الأطباق في المطاعم بعضها فوق بعض، وعند سحب طبق (من الأعلى) فإن هذا الطبق يمثل آخر طبق وُضع في المكدس. يظهر الشكل 11 ما يحدث عند تنفيذ سلسلة من تعليمات PUSH و POP.

ولا يغيبنّ عن الأذهان أن القيم تُسترجع في الترتيب المعكوس للدفع. وينبغي أيضاً أن يكون عدد تعليمات POP مساوياً لعدد تعليمات PUSH للمحافظة على توازن المكدس.

يفضل بعض المبرمجين دفع السجلات إلى المكدس في البرنامج الرئيسي بدلاً من دفعها في الإجرائية. وفي هذه الحالة، لا ندفع إلا السجلات التي نهتم بالحفاظ على قيمها، عند طلب الإجرائية. وتعاني

هذه الطريقة شيئاً من تعقيد البرنامج الرئيسي الذي يغدو مزدحماً بتعليمات PUSH و POP. ولذا يُفضل دفع السجلات واسترجاعها دوماً في الإجرائية.



الشكل 11: تأثير تنفيذ تعليمات PUSH و POP في المكدس.

6-6 تمرير المعاملات إلى/من الإجرائيات

نرغب عادةً، عند طلب إجرائية ما، أن نجعل بعض قيم المعطيات أو عناوين المتحولات في متناول الإجرائية. وبالمماثلة، نرغب غالباً في جعل بعض قيم المعطيات أو العناوين في متناول البرنامج الرئيسي. تُسمى هذه القيم الممررّة من وإلى الإجرائيات معاملات (أو محددات) الإجرائيات Procedure Parameters. توجد عدة طرائق لتمرير المعاملات Parameter Passing أهمها: التمرير بالسجلات، التمرير بمواقع الذاكرة، والتمرير بالمكدس.

لنبيّن بالمثال التالي الفروق بين طرائق التمرير.

تعريف المسألة وكتابة الخوارزمية:

نريد مثلاً تحويل عدد مرمّز بالترميزBCD مثل $(4569)_{BCD}$ إلى مكافئه بالترميز الست عشري. إن العدد المكافئ له هو: 11F4h.

توجد عدة طرائق لإجراء هذا التحويل ولكن أسهلها هو جداء كل رقم في هذا العدد بالمرتبة المقابلة له. فمثلاً يُكتب العدد السابق بالترميز الاثناني على النحو التالي:

4596 =(4*1000)+(5*100)+(9*10)+(6*1)

ولكن:

1000 = 3E8h
100 = 064h
10 = Ah
1 = 1h

ومنه:

4000 = 4*3E8 = FA0h
500 = 5*064 = 1F4h
90 = 9* 00Ah = 5Ah
6 = 6*1 = 6h

وبالجمع نجد: 11F4h.
فنضرب الآحاد بالعدد 6 والعشرات بـ 9 والمئات بـ 5 والآلاف بـ 4 ونجمع النتيجة فنحصل على قيمة العدد مرمزاً بالترميز الست عشري.

1-6-6 استخدام السجلات

يمكن، كحل أولي، تمرير العدد المراد تحويله عبر السجل AL إلى الإجرائية، وتُمرر النتيجة عبر السجل AL لأنها ستُرمز حتماً على 8 خانات. تبدأ الإجرائية بدفع سجل الحالة والسجلات الأخرى إلى المكدس، ونلاحظ أننا لا ندفع السجل AX إلى المكدس لأنه يحتوي على القيمة المراد تحويلها.

يبدأ التحويل بشحن العدد في السجل AH (لتوفير نسخة احتياطية)، وتُعزل الأوزان الدنيا منه وتُخزن في السجل BL. كما تُعزل من النسخة الأخرى للعدد الأوزان العليا وتُدوّر لتحلّ مكان الأوزان الدنيا. نضرب بعد ذلك القيمة الناتجة بالعدد 0Ah (أي العشرات) وتُخزن النتيجة في AX. ولما كانت النتيجة قابلة للترميز على 8 خانات، فإننا نهمل السجل AH ونجمع إلى السجل AL الأوزان الدنيا المخزّنة في BL. نحصل إذن في السجل AL على النتيجة النهائية.

تُسترجع في نهاية الإجرائية قيمة السجلات وتُنفذ تعليمة RET للعودة إلى البرنامج الرئيسي.

```
DATA_HERE: SEGMENT
BCD_INPUT DB ?              ; storage for BCD value
HEX_VALUE DB ?              ; storage for binary value
DATA_HERE: ENDS
```

```
CODE_HERE: SEGMENT WORD
ASSUME CS: CODE_HERE, DS: DATA_HERE, SS: STACK_HERE
    MOV AL, BCD_INPUT
    CALL BCD_HEX
    MOV HEX_VALUE, AL          ; store the result

; procedure BCD_HEX
; converts BCD numbers to HEX,
; uses registers to pass parameters to the procedure
; saves all resgisters except AH
BCD_HEX: PROC NEAR
    PUSHF; save flags
    PUSH BX
    PUSH CX
; start conversion
    MOV AH, AL                 ; save copy of BCD in AH
    AND AH, 0Fh                ; separate and save lower BCD digit
    MOV BL, AH
    AND AL, 0F0h               ; separate upper nibble
    MOV CL, 04                 ; move upper BCD digit to low
    ROR AL, CL                 ; nibble position for multiply
    MOV BH, 0Ah                ; load conversion factor in BH
    MUL BH                     ; upper BCD digit in AL * 0Ah in BH
                               ; result in AX
    ADD AL, BL                 ; add lower BCD to result of MUL
                               ; final result in AL
; end conversion, restore registers
    POP CX
    POP BX
    POPF
    RET
BCD_HEX: ENDP
CODE_HERE: ENDS

END
```

2-6-6 التمرير بمواقع الذاكرة

في بعض الحالات، قد لا تكفي السجلات لتمرير المعامِلات،

فيُستعاض عنها بالذاكرة. تُستخدم إذن مواقع الذاكرة لتمرير المعاملات الكثيرة العدد من وإلى الإجرائيات. ونجد فيما يلي نسخة جديدة من البرنامج المكتوب لتحويل الأعداد من الترميز BCD إلى الترميز الست عشري، وهو يستخدم مواقع الذاكرة لتمرير المعاملات بدلاً من السجلات.

```
DATA_HERE: SEGMENT
BCD_INPUT DB ?              ; storage for BCD value
HEX_VALUE DB ?              ; storage for binary value
DATA_HERE: ENDS

CODE_HERE: SEGMENT
ASSUME CS: CODE_HERE, DS: DATA_HERE, SS: STACK_HERE
    CALL BCD_HEX

; procedure BCD_HEX
; converts BCD numbers to HEX,
; uses memory locations to pass parameters to the procedure
; saves all resgisters except AH
BCD_HEX: PROC NEAR
    PUSH AX
    PUSHF                   ; save flags
    PUSH BX
    PUSH CX
; get BCD value from named memory location
    MOV AL, BCD_INPUT
    MOV AH, AL              ; save copy of BCD in AH
    AND AH, 0Fh             ; separate and save lower BCD digit
    MOV BL, AH
    AND AL, 0F0h            ; separate upper nibble
    MOV CL, 04              ; move upper BCD digit to low
    ROR AL, CL              ; nibble position for multiply
    MOV BH, 0Ah             ; load conversion factor in BH
    MUL BH                  ; upper BCD digit in AL * 0Ah in BH
                            ; result in AX
    ADD AL, BL              ; add lower BCD to result of MUL
                            ; final result in AL
```

```
; end conversion, restore registers
     MOV HEX_VALUE, AL
     POP CX
     POP BX
     POPF
     POP AX
     RET
BCD_HEX: ENDP
CODE_HERE: ENDS

END
```

ندفع داخل الإجرائية BCD2HEX كل السجلات والرايات المستخدمة في الإجرائية إلى المكدس، ثم ننقل العدد BCD من الذاكرة إلى السجل AL. وتتالى التعليمات بصورة مماثلة للنسخة السابقة إلى أن نصل إلى النقطة التي نريد فيها أن نمرّر النتيجة إلى البرنامج الرئيسي. عندئذٍ ننسخ محتوى السجل داخل الذاكرة، ونسترجع قيم الرايات والسجلات ثانيةً قبل تنفيذ التعليمة RET.

لهذه الطريقة بعض المحاذير، وهي أن الإجرائية تصل دوماً إلى الموقع BCDInput لتحصل منه على معطيات الدخل، وتضع النتيجة في الموقع HexValue. ومن ثَم لن نستطيع استخدام الإجرائية ذاتها لتحويل عدد BCD إذا كان مخزّناً في مكان آخر من الذاكرة.

3-6-6 التمرير بواسطة المؤشرات

يمكن لتخطي المشكلة السابقة، وهي استخدام أسماء المواقع مباشرةً في الإجرائية، أن نمرّر مؤشراً إلى موقع المعطيات. يصف البرنامج التالي طريقة لذلك.

```
BCD_INPUT DB ?        ; storage for BCD value
HEX_VALUE DB ?        ; storage for binary value
DATA_HERE: ENDS
```

```
CODE_HERE: SEGMENT
ASSUME CS: CODE_HERE, DS: DATA_HERE, SS: STACK_HERE
; put pointer to BCD in SI and pointer to HEX storage in DI
    MOV SI, OFFSET BCD_INPUT
    MOV DI, OFFSET HEX_VALUE
    CALL BCD_HEX

; procedure BCD_HEX
; converts BCD numbers to HEX,
; uses pointers to pass parameters to the procedure
; saves all resgisters except AH
BCD_HEX: PROC NEAR
    PUSH AX
     PUSHF                      ; save flags
     PUSH BX
     PUSH CX
; get BCD value from named memory location
    MOV AL, [SI]
; do conversion
    MOV AH, AL                  ; save copy of BCD in AH
    AND AH, 0Fh                 ; separate and save lower BCD digit
    MOV BL, AH
    AND AL, 0F0h                ; separate upper nibble
    MOV CL, 04                  ; move upper BCD digit to low
    ROR AL, CL                  ; nibble position for multiply
    MOV BH, 0Ah                 ; load conversion factor in BH
    MUL BH                      ; upper BCD digit in AL * 0Ah in BH
                                ; result in AX
    ADD AL, BL                  ; add lower BCD to result of MUL
    MOV [DI], AL                ; move hex value result in AL
                                ; to DS location pointed to by DI
    POP CX                      ; restore registers and flags
    POP BX
    POPF
    POP AX
    RET
BCD_HEX: ENDP
CODE_HERE: ENDS
END
```

نستخدم قبل طلب الإجرائية في البرنامج الرئيسي التعليمة MOV SI, OFFSET BCD_INPUT لكي يؤشر السجل SI على موقع الذاكرة BCD_INPUT، ونستخدم أيضاً التعليمة التالية: MOV DI, OFFSET HEX_Value ليؤشر السجل DI على موقع الذاكرة HEX_Value. يكفي عندئذٍ استخدام التعليمة MOV AL, [SI] داخل الإجرائية لنقل محتوى الموقع المؤشر عليه بالسجل SI إلى السجل AL. وكذلك عند تخزين النتيجة، تُستخدم التعليمة MOV [DI], AL لننسخ محتوى السجل AL إلى موقع الذاكرة المؤشر عليه بـ DI.

يُعد هذا النهج، الذي يستخدم مزيجاً من السجلات والذاكرة، أكثر النهوج كفاءة وتنوعاً لأنه يسمح للمبرمج أن يمرّر مؤشرات إلى أي موقع في الذاكرة، كما يسمح بتمرير مؤشرات إلى جداول أو مصفوفات.

ويستطيع المبرمج -إن لم يرغب في تمرير المؤشرات المخزّنة في سجلات المعالج مثل SI و DI- أن يخزّن هذه المؤشرات في مواقع الذاكرة. وفي هذه الحالة، تجلب الإجرائية أولاً المؤشرات من الذاكرة، ثم تستخدمها لجلب المعطيات المطلوبة. ولكن عندما يتشارك عدة مستخدمين في الموارد ذاتها، فإننا غالباً ما نلجأ إلى تمرير المعاملات عبر المكدس.

4-6-6 تمرير المعاملات عبر المكدس

يجب لتمرير المعاملات إلى إجرائية ما عبر المكدس، أن ندفعها أولاً إلى المكدس في مكان ما في البرنامج الرئيسي قبل طلب الإجرائية. ثم تُقرأ في الإجرائية هذه المعاملات من المكدس. وبالطريقة ذاتها، يجب، عند تمرير نتائج الإجرائية إلى البرنامج الرئيسي، على الإجرائية أن تكتبها في المكدس. ثم يقوم البرنامج الرئيسي بقراءتها من المكدس، وسيظهر مثال بسيط آلية العمل المذكورة .

```
DATA_HERE: SEGMENT
BCD_INPUT DB ?                    ; storage for BCD value
HEX_VALUE DB ?                    ; storage for binary value
DATA_HERE: ENDS

CODE_HERE: SEGMENT
ASSUME CS: CODE_HERE, DS: DATA_HERE, SS: STACK_HERE
PUSH AX                           ; put BCD_VALUE on stack
CALL BCD_HEX                      ; convert to hexadecimal
; program continues here with result of conversion on the top of stack

; procedure BCD_HEX
; converts BCD numbers to HEX,
; uses stack to pass parameters to the procedure
; saves all resgisters except AH
BCD_HEX: PROC NEAR
    PUSH AX
    PUSHF                         ; save flags
    PUSH BX
    PUSH CX
    PUSH BP
    MOV BP, SP                    ; copy SP into BP
    MOV AX, [BP+12]               ; copy BCD from stack to AX
; do conversion
    MOV AH, AL                    ; save copy of BCD in AH
    AND AH, 0Fh                   ; separate and save lower BCD digit
    MOV BL, AH
    AND AL, 0F0h                  ; separate upper nibble
    MOV CL, 04                    ; move upper BCD digit to low
    ROR AL, CL                    ; nibble position for multiply
    MOV BH, 0Ah                   ; load conversion factor in BH
    MUL BH                        ; upper BCD digit in AL * 0Ah in BH
                                  ; result in AX
    ADD AL, BL                    ; add lower BCD to result of MUL
                                  ; final result in AL
; end of conversion, move hex value from AL to location onto the stack
    MOV [BP+12], AX
    POP BX                        ; restore registers
    POP CX                        ; restore flags and return
```

```
    POP BX
    POPF
    POP AX
    RET
BCD_HEX: ENDP
CODE_HERE: ENDS

END
```

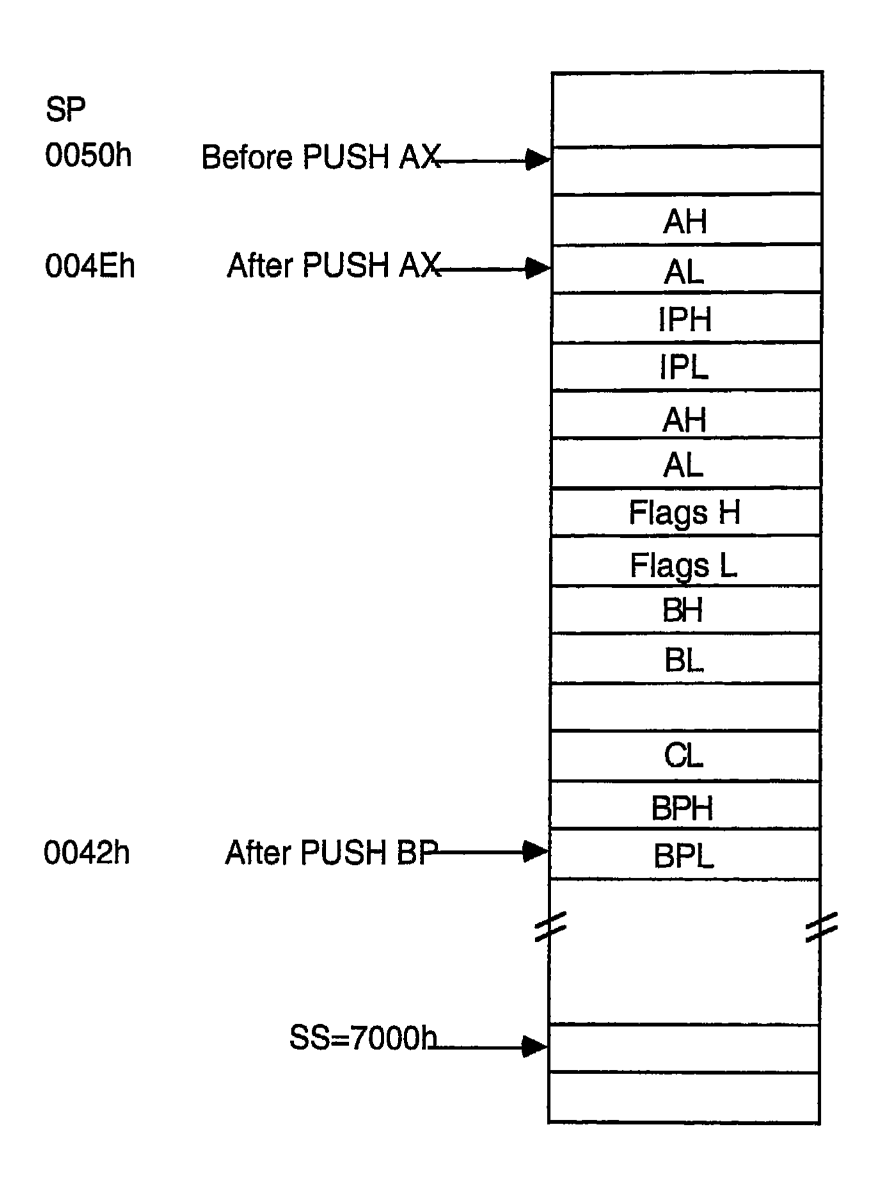
يعتمد البرنامج السابق طريقة تمرير المحددات عبر المكدس. نفترض للسهولة أن استهلال المكدس قد حصل في بداية البرنامج، ونفترض أيضاً أن العدد المراد تحويله موجود في السجل AL. يقوم البرنامج عندئذٍ بدفع السجل AX إلى المكدس PUSH AX قبل طلب الإجرائية.

عند تنفيذ تعليمة CALL يُنقص مؤشر المكدس بمقدار 2، ويُنسخ عنوان العودة في المكدس. تقوم الإجرائية بعد ذلك بحفظ السجلات والرايات المستخدمة بواسطة التعليمات PUSH الموجودة في بداية الإجرائية.

لنلقِ، قبل المتابعة، نظرة إلى محتويات المكدس (الشكل 12).

نلاحظ أن موقع العدد المراد تحويله يعلو موقع عنوان العودة. والسؤال هو كيف نستطيع الوصول إلى هذا العدد داخل الإجرائية علماً بأن مؤشر المكدس أصبح يؤشر على موقع تخزين السجل BP؟

تنص إحدى الطرائق الممكنة على جمع العدد 12 إلى مؤشر المكدس بحيث يؤشر السجل SP إلى العدد المطلوب. ثم يُشحن هذا العدد في السجل AX بتعليمة POP AX. ولكن هناك عدة عوامل -سنوضحها لاحقاً- تحذرنا من تغيير مؤشر المكدس.



الشكل 12: محتوى المكدس أثناء التنفيذ.

ذكرنا سابقاً أن السجل BP يؤدي دور مؤشر ثانٍ على المكدس، لأنه يمثل الانزياح الذي يُضاف إلى سجل قطاع المكدس SS لتوليد عنوان

حقيقي للذاكرة. ونستفيد من ذلك للحصول على العدد المراد تحويله.

ففي الإجرائية، ننسخ محتويات سجل مؤشر المكدس SP إلى السجل BP بالتعليمة MOV BP, SP، فيؤشر عندها BP على الموقع ذاته الذي يؤشر عليه مؤشر المكدس SP. ثم نستخدم التعليمة MOV AX, [BP+12] لننسخ العدد المطلوب تحويله إلى السجل AX. ويضيف المعالج، عند تنفيذه لهذه التعليمة، القيمة 12 إلى محتوى السجل BP، ومن ثَم يصبح العنوان الفعلي هو 004Eh. نلاحظ أن هذه التعليمة لا تخزّن قيمة الانزياح الجديدة في السجل BP، ومن ثَم، يبقى بالإمكان استخدامه للتأشير على مواقع أخرى بتحديد الانزياح المناسب فقط.

بعد تحويل العدد الى النظام الست عشري نستطيع تخزين النتيجة في المكدس بالطريقة ذاتها. نستعمل إذن السجل BP في التعليمة التالية: MOV [BP+12], AX، فيُخزن محتوى السجل AX في عنوان يبعد بمقدار 12 ثمانيّة عن مكان تأشير السجل BP. وهذا هو بالطبع المكان ذاته الذي استخدمناه في الحصول على المعطيات.

أما النتيجة فتُسترجع في البرنامج الرئيسي من المكدس الذي استرجعت كافة محتوياته عدا النتيجة المخزّنة بالتعليمة POP CX.

ومن الضروري، عند استخدام المكدس في تمرير المعاملات، متابعة الحركة فيه من حيث الدفع والاسترجاع. وتُعدّ المخططات المماثلة لتلك الموضحة في الشكل 12 مساعدة جداً في ذلك.

ويجب الانتباه، عند استخدام المكدس، إلى مشكلة فيضان المكدس Stack Overflow التي تحدث عندما يمتلئ قطاع المكدس ويفيض إلى قطاع آخر. وهذا ما قد يحدث بسبب تخزين كمية كبيرة من المعطيات في المكدس أو نتيجة خطأ برمجي. فإذا لم تُسترجع كافة المعطيات المخزّنة في المكدس داخل الإجرائية، أي إذا لم يعد مؤشر المكدس SP إلى الموقع الذي كان عليه قبل طلب الإجرائية، فإنه سيفيض نتيجةً للطلب المتكرر ليؤشر على مكان خارج القطاع. ويؤدي ذلك إلى حدوث أخطاء قد تسبب توقف كامل النظام الحاسوبي.

لعلاج هذه المشكلة، يجب استخدام مخططات المكدس، التي تساعد على الحفاظ على توازن المكدس. ويجب أيضاً، كقاعدة عامة، أن يكون عدد التعليمات POP مساوياً لعدد التعليمات PUSH.

7-6 طلب الإجرائيات البعيدة

نصف إجرائيةً ما بأنها بعيدة Far إذا وقعت في قطاع مغاير للقطاع الذي يحوي تعليمة الطلب CALL. وفي هذه الحالة، يشحن المعالج سجل قطاع البرنامج CS، ومؤشر التعليمات IP بالقيم المناسبة للانتقال إلى الإجرائية البعيدة. ونميّز هنا بين حالتين:

- حالة الإجرائية البعيدة المكتوبة في ملف البرنامج الرئيسي.
- الإجرائية البعيدة المكتوبة في ملف آخر.

• الحالة الأولى:

سنفترض أن الإجرائيات والبرنامج الرئيسي موجودة في ملف واحد. ولكن البرنامج الرئيسي موجود في قطاع منطقي معين، وأن الإجرائيات مخزّنة في قطاع آخر.

نعرض فيما يلي أجزاءً من هذا البرنامج:

```
Code_here: SEGMENT
ASSUME: CS: Code_here, DS: Data_here, SS: Stack_here
...
CALL Multply-32
...
Code_here: ENDS

Procedure_here: SEGMENT
Multiply_32: PROC FAR
ASSUME CS: Procedure_here
...
Multiply_32: ENDP
Precedure_here: ENDS
```

في هذا المثـال، تقع تعليـمـات البـرنامج الرئيـسي في القطاع Code_here، وتقع تعليـمـات الإجرائيـة في القطاع Procedure_here. ولما كانت الإجرائية موجودة في قطاع مغاير للتعليمة CALL، وجب تغيير محتوى السجلين CS و IP للوصول إليه. ولإعلام المجمِّع بأن الإجرائية بعيدة، أضيف التوجيه FAR إلى تعريف الإجرائية Multiply_32. وعندئذٍ، يرمّز المجمّع تعليمات CALL كتعليمات طلب بعيد.

يُستخدم التوجيه ASSUME لشحن السجل CS بالقيمة الصحيحة الموافقة للقطاع. فالسطر ASSUME CS: Code_here يدفع المجمِّع إلى حساب الانزياحات بدءاً من العنوان Code_here.

وصفوة القول، عندما تكون الإجرائية بعيدة ينبغي تصريحه بالتوجيه FAR. كما يجب وضع التوجيه ASSUME في الإجرائية لمعرفة العنوان القاعدي الذي سيستخدمه المجمِّع حين الترجمة.

- الحالة الثانية:

عندما يكون البرنامج المطلوب تحقيقه كبيراً جداً، من الأفضل كتابته في عدة ملفات بحيث يحتوي كل منها على جزء من البرنامج. و يمكن كلَّ جزء أن يُكتب مستقلاً و أن يُجمَّع و يُختبر وحده. تُجمع بعد ذلك الأجزاء معاً بواسطة برمجية متخصصة تسمى محرِّر الروابط Linker.

ولكي يستطيع محرِّر الروابط الوصول الى الأجزاء المكتوبة في ملفات متعددة ينبغي تزويد المجمِّع بتوجيهات أربع: يجب أن يحتوي ملف البرنامج الرئيسي على التوجيه EXTERN للدلالة على أن الإجرائيات موجودة في ملفات خارجية. ويجب أن يحتوي البرنامج الرئيسي على التوجيه PUBLIC للدلالة على أن بعض أسماء اللصاقات ستُطلب من الخارج. وعلى نحو مماثل، يجب أن تُصرح أسماء اللصاقات الخارجية في ملف الإجرائيات بالتوجيه EXTERN، وأسماء الإجرائيات المستخدمة في ملفات خارجية بالتوجيه PUBLIC

7 المقاطعات

تسمح معظم المعالجات الصغرية بـ «مقاطعة» تنفيذ البرنامج بناءً على إشارات خارجية أو على تعليمات خاصة بها. فعند مقاطعة معالج ما، فإنه يتوقف عن تنفيذ برنامجه الحالي، ويستدعي إجرائية تخدّم هذه المقاطعة. وبعد الانتهاء من تنفيذ هذه الإجرائية، يعود المعالج إلى البرنامج السابق، ويستأنف التنفيذ من النقطة التي توقف عندها.

سنعرض في هذه الفقرة آلية استجابة المعالجات 8086 للمقاطعات وطريقة كتابة إجرائيات خدمة المقاطعة Interrupt Service.

1-7 وصف مقاطعات المعالج 8086

يمكن أن يُقاطع عمل المعالج 8086 بأحد مصادر ثلاثة، وهي:

- إشارة خارجية تُطبق على المربط NMI (وهي المقاطعة غير القابلة للحجب)، أو على المربط INTR (مدخل المقاطعة القابلة للحجب). وتسمى المقاطعة الناجمة عن مثل هذه الإشارات مقاطعة الكيان الصلب Hardware Interrupt.
- تنفيذ تعليمة المقاطعة INT، وتدعى المقاطعة المبرمَجة Software Interrupt.
- تحقق شرط معين نتيجة تنفيذ تعليمة ما في المعالج. ومثال ذلك، المقاطعة الناتجة من القسمة على 0؛ إذ يتوقف البرنامج تلقائياً عندما نحاول قسمة عدد ما على الصفر. وتُسمى هذه المقاطعات الشرطية بالمقاطعات البرمجية أيضاً.

يفحص المعالج في نهاية كل دور تعليمة Instruction Cycle حالة المقاطعات، فإذا وجد أن هناك مقاطعة ما، فإنه يستجيب لها بالخطوات التالية:

1 ينقص مؤشر المكدس بمقدار 2، ويدفع بسجل الرايات إلى المكدس.

2 يلغي تأهيل المدخل INTR بمحو راية المقاطعة في سجل الرايات.

3 يضع صفراً في راية التنفيذ الخطوي TRAP المحتواة في سجل الرايات.

4 ينقص مؤشر المكدس بمقدار 2، ويدفع محتوى سجل قطاع البرنامج الحالي إلى المكدس.

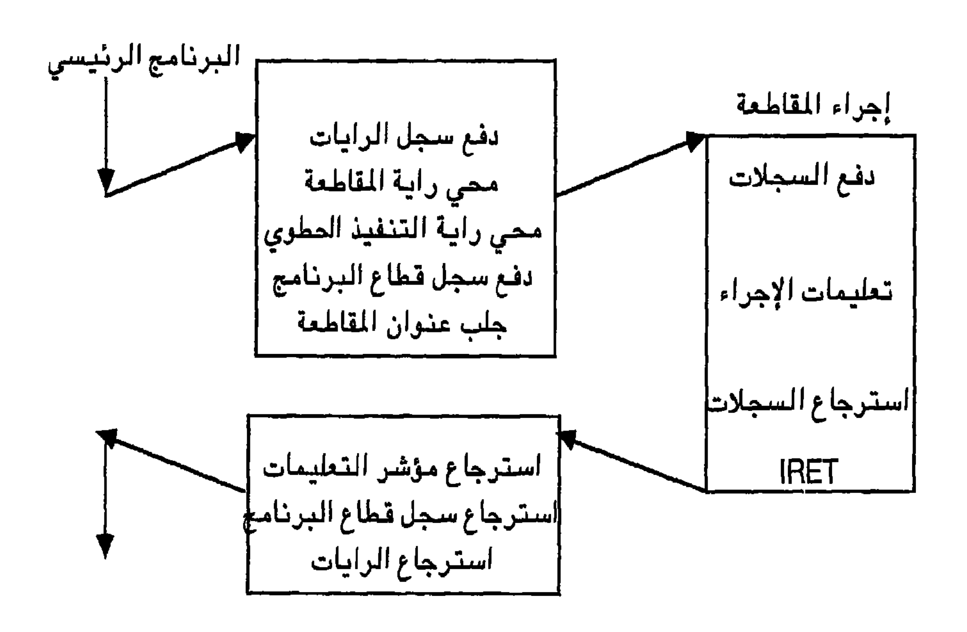
5 ينقص مؤشر المكدس مرة ثانية بمقدار 2، ويدفع محتوى مؤشر التعليمات الحالي إلى المكدس.

6 يجري المعالج قفزاً غير مباشر إلى بداية الإجرائية المكتوبة لخدمة المقاطعة.

بكلمات أخرى، يدفع المعالج سجل الرايات إلى المكدس، ويلغي تأهيل راية التنفيذ الخطوي، ويحجب مدخل المقاطعة INTR، ثم يقوم باستدعاء إجرائية خدمة المقاطعة استدعاء بعيداً وغير مباشر. يظهر الشكل 13 هذه المراحل.

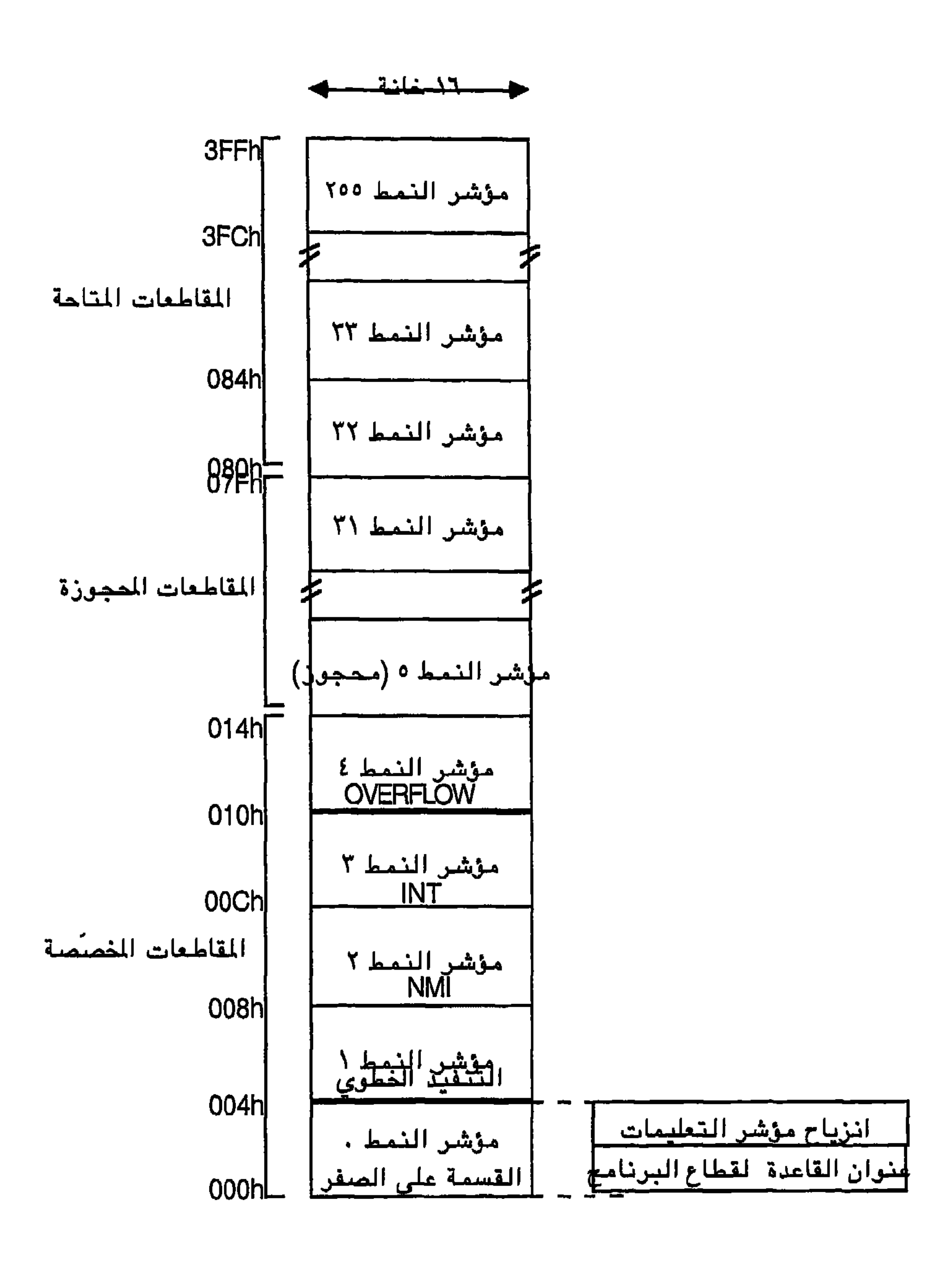
نلاحظ أن تعليمة RET في نهاية إجرائية خدمة المقاطعة تعيد السيطرة إلى البرنامج الرئيسي. ونذّكر بأن المعالج 8086، عندما يطلب إجرائية بصورة بعيدة وغير مباشرة، يضع قيمة جديدة في سجل قطاع البرنامج CS وقيمة جديدة في مؤشر التعليمات IP؛ ويحصل المعالج 8086 في تلك الحالة على القيم الجديدة لـ CS و IP من أربعة مواقع متلاحقة في الذاكرة. وبالمماثلة، عندما يستجيب المعالج 8086 إلى إحدى المقاطعات فإنه يتوجه إلى الذاكرة في بداية إجرائية المقاطعة للحصول على قيمة السجلين CS و IP.

يحتوي أي نظام حاسوبي مبني على المعالج 8086 في ذاكرته المحصورة بين العنوان 0h والعنوان 03FFh على جدول يخزّن عناوين البداية لبرامج خدمة المقاطعات.



الشكل 13: مراحل الاستجابة للمقاطعة.

ولما كانت كل إجرائية مقاطعة تتطلب تخزين أربع ثمانيّات لعنوان السجل CS وIP، فإن الجدول يستطيع تخزين العناوين المتعلقة بـ 256 إجرائية خدمة مقاطعة. ويُسمى عنوان بداية كل إجرائية متجه المقاطعة Interrupt Vector أو مؤشر المقاطعة Interrupt Pointer، كما يُسمى الجدول جدول متجهات المقاطعة Interrupt Vector Table. يظهر الشكل 14 توضع متجهات المقاطعة في الذاكرة. ويشار إلى كل كلمة مزدوجة (أي أربع ثمانيّات) بعدد يبدأ بالصفر وينتهي بالقيمة 255. ويدعى هذا العدد نوع المقاطعة Interrupt Type.



الشكل 14: توضع متجهات المقاطعة في الذاكرة.

من ناحية أخرى، تُخصص المقاطعات الخمس الدنيا لعمليات معينة في المعالج، مثل القسمة على الصفر والمقاطعة غير القابلة للحجب. أما الأنواع من 5 إلى 31 فهي غير معرّفة وتُستخدم في معالجات الأجيال المتقدمة. و يستطيع المستثمر أن يستخدم أنواع المقاطعة المحصورة بين 32 و 255 كمقاطعات برمجية أو مقاطعات الكيان الصلب.

عندما يستجيب المعالج 8086 لمقاطعة ما، فإنه يذهب تلقائياً إلى الموقع المحدّد بتلك المقاطعة في جدول متجهات المقاطعة، ويحصل منه على عنوان البداية لإجرائية خدمة المقاطعة. ولا تجري عملية شحن عناوين البداية في جدول متجهات المقاطعة تلقائياً، بل ينبغي للمستثمر أن يشحن، في بداية برنامجه، القيم المناسبة في ذلك الجدول.

لنعرض الآن مثالاً تفصيلياً على آلية عمل إحدى المقاطعات.

<u>مثال</u>: استجابة المعالج 8086 لمقاطعة من النوع 0

تحدث المقاطعة من النوع 0 عند تعرض المعالج لقسمة عدد ما على القيمة 0، ونبيّن فيما يلي طريقة كتابة إجرائية الخدمة لهذه المقاطعة.

يوجد في المعالج تعليمتان للقسمة هما DIV و IDIV. تسمح التعليمة الأولى DIV بقسمة عدد بلا إشارة مرمّز على 16 خانة ومخزّن في السجل AX، على عدد بلا إشارة مرمّز على 8 خانات ومخزّن في سجل ما من المعالج أو في موقع من الذاكرة. وبعد تنفيذ القسمة، يحتوي السجل AL على القسم الصحيح من خارج القسمة، في حين سيُخزن باقي القسمة في السجل AH.

يمكن بواسطة هذه التعليمة قسمة عدد بلا اشارة ذي 32 خانة ومخزّن في السجلين DX و AX على عدد ذي 16 خانة ومخزّن في سجل ما أو في موقعين متتاليين من الذاكرة. وعندئذٍ، يُخزن القسم

الصحيح من خارج القسمة في السجل AX، أما باقي القسمة فيُخزن في السجل DX. وتعمل التعليمة IDIV عمل التعليمة السابقة ولكنها تسمح بقسمة أعداد ذوات اشارة. عندما يكون المقسوم عليه مساوياً للصفر تصبح عملية القسمة بلا معنى، والخارج هو اللانهاية. وهذا مايتعذر ترميزه في السجلين AX وDX، فيتعرض عندئذٍ المعالج لمقاطعة من النوع 0.

ينقص المعالج، مستجيباً لهذه المقاطعة، مؤشر المكدس بمقدار 2، ويدفع سجل الرايات إلى المكدس، ثم يكتب القيمة 0 في راية التنفيذ الخطوي TF وراية المقاطعة IF لإلغاء عملهما. يخزن المعالج بعد ذلك عنوان العودة في المكدس. ولإجراء ذلك ينقص مؤشر المكدس بمقدار 2 أيضاً، ويدفع محتوى السجل CS إلى المكدس، ثم ينقص المكدس ثانيةً بمقدار 2، ويدفع محتوى مؤشر التعليمات IP إلى المكدس.

يحصل المعالج بعد ذلك من جدول متجهات المقاطعة على عنوان البداية لإجرائية المقاطعة من النوع 0، فيشحن محتوى الموقع 02h والموقع 03h في السجل CS، ومحتوى الموقعين 00h و 01h في السجل IP.

بعد ذلك، يبدأ المعالج بتنفيذ التعليمة الأولى في إجرائية خدمة المقاطعة. عندما يصل المعالج إلى نهاية الإجرائية يصادف تعليمة العودة IRET. فيسترجع القيم المخزّنة في المكدس ويشحنها في السجلين CS و IP ويزيد مؤشر المكدس في كل مرة بمقدار 2. كما يسترجع المعالج سجل الرايات من المكدس ويزيد مؤشر المكدس بمقدار 2. وكما ذكرنا سابقاً، يلغي المعالج عند مقاطعته تأثير الرايتين TF و IF. ولذا، فإنه يكتب فيهما، عند الانتهاء من إجرائية المقاطعة، قيمهما قبل المقاطعة. وبمعنى آخر، فإن التعليمة IRET تعيد السيطرة إلى البرنامج الرئيسي وتعيد إلى الرايتين IF و TF قيمهما قبل المقاطعة.

تعريف المسألة وكتابة الخوارزمية:

نريد هنا قسمة أربع كلمات مخزّنة في الذاكرة على قيمة مرمّزة على ثمانيّة واحدة، لنحصل في النتيجة على أربع ثمانيّات تمثل القيم النهائية المطلوبة. فإذا كانت نتيجة القسمة غير صالحة (أي إن القيمة كبيرة جداً بحيث يصعب ترميزها في سجل المعالج)، فإننا نكتب القيمة 0 في النتيجة النهائية. تُكتب إذن الخوارزمية على النحو التالي:

- البرنامج الرئيسي:

استهلال؛

كرّر مايلي

احصل على القيمة؛

أجرِ عملية القسمة؛

إذا كانت نتيجة القسمة صالحة افعل

خزّن النتيجة؛

وإلا

خزّن القيمة 0؛

إلى أن تنتهي قيم الدخل؛

- جرائية خدمة المقاطعة:

خزّن السجلات؛

رفع راية الخطأ؛

استرجع السجلات؛

عُد إلى البرنامج الرئيسي؛

يقوم إذن البرنامج الرئيسي بقراءة القيم من الذاكرة، وإجراء عمليات القسمة، ثم تخزين النتيجة إذا كانت القسمة صالحة. وإلا،

فإنه يخزّن القيمة 0 بدلاً منها. أما إجرائية خدمة المقاطعة فهي تقوم، بعد تخزين السجلات، برفع راية تدل على الخطأ. إن هذه الراية لا تنتمي إلى سجل رايات المعالج 8086، ولكن يُقصد بالراية هنا كتابة قيمة معينة في موقع من الذاكرة للدلالة على حدوث الخطأ. تسترجع بعدئذٍ إجرائية خدمة المقاطعة السجلات المخزّنة في المكدس وتعيد السيطرة إلى البرنامج الرئيسي.

تُفحص نتيجة القسمة في البرنامج الرئيسي بعد تنفيذ القسمة. ويجري ذلك بفحص قيمة راية الخطأ. فإذا كانت راية الخطأ مساوية للصفر، فهذا يدل على أن القسمة صالحة، ومن ثم فإننا نكتب خارج القسمة في الذاكرة. وإلا، فإننا نخزّن القيمة 0 في جدول النتائج. ففي هذه الحالة تحتوي راية الخطأ على قيمة غير معدومة ناتجة من حدوث مقاطعة القسمة على صفر. ويتكرر تنفيذ عمليات القسمة إلى أن تنتهي قيم الدخل.

كتابة قائمة الاستهلال:

ينبغي بعد كتابة خوارزمية البرنامج كتابة قائمة الاستهلال. ففي هذا المثال، تتألف القائمة مما يلي:

- جدول متجهات المقاطعة: يجب وضع عنوان البداية لإجرائية خدمة المقاطعة في المواقع من 0 إلى 3؛
- قطاع المعطيات: الذي تُخزّن فيه قيم الدخل والقيم النهائية؛
- سجل قطاع المعطيات DS: الذي يؤشر على العنوان القاعدي لقطاع المعطيات ويحتوي على قيم الدخل؛
- المكدس: الذي يُستخدم لتخزين عنوان العودة؛
- سجل قطاع المكدس SS ومؤشر المكدس SP؛
- مؤشر إلى بداية جدول النتائج النهائية؛
- عداد لعدد قيم الدخل المُعالَجة؛
- مؤشر إلى بداية قيم الدخل.

البرنامج بلغة المجمّع:

```
DATA_HERE: SEGMENT WORD PUBLIC
INPUT_VALUES DW 0035h, 0855h, 2011h, 1359h
SCALED_VALUES DB 4 DUP(0)
SCALE_FACTOR DB 09h
BAD_DIV_FLAG DB 0
DATA_HERE ENDS

STACK_HERE SEGMENT STACK DW 100 DUP(0)
                                   ; set up stack of 100 words
TOP_STACK LABEL WORD               ; pointer to top of stack
STACK_HERE ENDS

PUBLIC BAD_DIV_FLAG                ; make flag available to
                                   ; other modules

INT_PROC_HERE SEGMENT WORD PULIC
EXTRN BAD_DIV: FAR                 ; let assembler know
                                   ; procedure of BAD_DIV is
                                   ; in other assembly module

CODE_HERE SEGMENT WORD PUBLIC
ASSUME CS: CODE_HERE, DS: DATA_HERE, SS:STACK_HERE
START:
    MOV AX, STACK_HERE             ; initialize stack segment register
    MOV SS, AX
    MOV SP, OFFSET TOP_STACK       ; initialize stack pointer
    MOV AX, DATA_HERE              ; initialize data segment register
    MOV DS, AX
; store the address for the BAD_DIV routine at address 0000: 0000
; address 0000-0003 is where type 0 interrupt gets interrupt
; service procedure address, CS at 0002 and 0003, IP at 0000 and 0001
    MOV AX, 0000
    MOV ES, AX
    MOV WORD PTR ES:0002, SEG BAD_DIV
    MOV WORD PTR ES:0000, OFFSET BAD_DIV
```

```
    MOV SI, OFFSET INPUT_VALUES
                                ; initialize pointer for input values
    MOV BX, OFFSET SCALED_VALUES
                                ; point BX at start of result array
    MOV CX, 0004h               ; initialize data value
                                ; to AX for divide
    NEXT: MOV AX, [SI]          ; Bring a value to AX for divide
    DIV SCALE_FACTOR            ; divide by Scale factor
    CMP BAD_DIV_FLAG, 01h       ; check if divide produced
                                ; invalid result
    JNE OK                      ; NO: go save scaled value
    MOV BYTE PTR[BX], 00        ; YES: load a 0 as scaled value
    JMP SKIP
OK:
    MOV [BX], AL                ; save scaled value
SKIP:
    MOV BAD_DIV_FLAG, 0         ; reset BAD_DIV_FLAG
                                ; before doing next
    ADD SI, 02h                 ; point at location of
                                ; next input value
    INC BX                      ; point at location for next result
    LOOP NEXT                   ; repeat until all values done
STOP:
    NOP
CODE_HERE: ENDS

END START

; service divide by zero interrupt
DATA_HERE SEGMENT WORD PUBLIC
EXTRN BAD_DIV_FLAG: BYTE        ; let assembler know
                                ; BAD_DIV_FLAG
DATA_HERE ENDS                  ; is in another assembly module

PUBLIC BAD_DIV                  ; make procedure BAD_DIV
                                ; available to other
                                ; assembly modules
```

```
INT_PROC_HERE SEGMENT WORD PUBLIC
                                 ; set up a segment for all interrupt
                                 ; service procedures
BAD_DIV PROC FAR                 ; procedure for type 0 interrupt
ASSUME CS: INT_PROC_HERE, DS: DATA_HERE
    PUSH AX                      ; save AX of interrupted program
    PUSH DS                      ; save DS of interrupted program
    MOV AX, DATA_HERE            ; load DS value needed here
    MOV DS, AX
    MOV BAD_DIV_FLAG, 01         ; set LSB of BAD_DIV_FLAG
                                 ; byte
    POP DS                       ; restore DS of
                                 ; interrupted program
    POP AX                       ; restore AX of
                                 ; interrupted program
    IRET                         ; return to next instruction in
                                 ; interrupted program
BAD_DIV ENDP
INT_PROC_HERE ENDS

END
```

كُتب البرنامج الرئيسي في ملف مستقل عن الملف الذي كُتبت فيه إجرائية خدمة المقاطعة. وهذا ما يسوغ استخدام التوجيهين EXTRN و PUBLIC.

نصرّح في بداية البرنامج الرئيسي بوجود قطاع يُسمى Data_Here يحوي المعطيات التي سيعالجها البرنامج. ويطلب التوجيه WORD من المجمِّع بدء هذا القطاع في أول عنوان زوجي متاح.

أما التوجيه PUBLIC فإنه يعرّف القطاع عمومياً، أي يمكن وحداتٍ برمجيةً مكتوبةً في ملفات خارجية الوصولُ إليه. ولما كانت كلمات الدخل قيماً مرمزة على 16 خانة، فإننا نستخدم التوجيه DW لتصريح هذه القيم. أما قيم النتائج فهي ثمانيّات، ولذا نستخدم التوجيه DB لحجز أربعة مواقع ذاكرة لها. إضافة إلى ذلك، تُشحن تلك المواقع الأربعة بقيمة الصفر بواسطة التوجيه DUP(0).

أما العبارة Scale_Factor DB 09h فهي تحجز ثمانيّة خاصة لتخزين العدد الذي سيقسم كافة قيم الدخل. وسبب استخدام DB بدلاً من EQU في تعريف المتحول Scale_Factor هو أن القيمة المحدَدة بـ DB تكون موجودة في الذاكرة الحية RAM، ولذا يمكن تغييرها ديناميكياً في البرنامج. أما التوجيه EQU فإن يعرّف قيمة لايمكن تغييرها إلا بإعادة تجميع البرنامج من جديد.

ثم يقوم البرنامج الرئيسي بتعريف مكدس لتخزين عنوان العودة والمعاملات الممرَرة من/إلى الإجرائية. ولذا يُعرف القطاع Stack_Here بوضع مؤشر على قمة المكدس بالعبارة Top_Stack Label WORD. تُستخدم هذه اللصاقة عند الاستهلال ليشير مؤشر المكدس نحو الموقع الذي يلي قمة المكدس مباشرة.

يلي ذلك تصريح راية الخطأ BAD_DIV_Flag، والمعرَّفة بالتوجيه PUBLIC، لكي تستطيع الوحدات البرمجية المكتوبة في ملفات خارجية من الوصول إليه. ومن أجل ذلك، يجب أن تُعرّف هذه الراية في تلك الوحدات باستخدام التوجيه EXTERN، للدلالة على أن هذا المتحول موجود في ملف آخر.

وبكلمات أخرى، يسمح التوجيه PUBLIC بتصدير أسماء اللصاقة نحو الوحدات البرمجية الخارجية، في حين يسمح التوجيه EXTERN باستيرادها.

توجه العبارتان INT_PROC_here SEGMENT WORD PUBLIC و INT_PROC_here ENDS المجمِّع لتعريف الإجرائية BAD_DIV في قطاع اسمه INT_PROC_here.

يُعرّف بعد ذلك قطاع البرنامج بالعبارة CODE_HERE SEGMENT WORD PUBLIC، وتشير الكلمة PUBLIC إلى أن القطاع يمكن جمعه مع قطاعات لها الاسم ذاته وموجودة في وحدات برمجية خارجية.

ويفيد التوجيه ASSUME في تحديد القطاعات المنطقية المستخدمة، كقطاع البرنامج وقطاع المعطيات والمكدس. ثم تأتي

تعليمات الاستهلال التقليدية لسجل قطاع المكدس SS ومؤشر المكدس SP وسجل قطاع المعطيات DS. تشحن التعليمات الأربع التالية عنوان الإجرائية BAD_DIV في الموقع المناسب في جدول متجهات المقاطعة، إذ نستخدم هنا المقاطعة من النوع 0.

يجري بعد ذلك وضع قيمة ابتدائية مناسبة في السجل SI ليؤشر على بداية قيم الدخل، والسجل BX ليؤشر نحو بداية جدول النتائج، ويستخدم أيضاً السجل CX كعداد لقيم الدخل المُعالجة، فعندما يصبح CX=0 فهذا يعني أن جميع القيم قد قُسمت.

ثم تبدأ عمليات المعالجة، إذ تُقرأ كلمة دخل، وتُوضع في السجل AX، ثم تُقسم على القيمة Scale_Factor، وتصبح نتيجة القسمة موجودة في السجل AL. فإذا كانت النتيجة كبيرة جداً بحيث يتعذر تخزينها في ذلك السجل، تولدت مقاطعة من النوع 0، وهذا ما يدفع المعالج إلى القفز نحو اجراء خدمة المقاطعة، بعد الحصول على عنوانه من جدول متجهات المقاطعة (المواقع من 0 إلى 3). ومن ثَم، فالمعالج يقفز نحو الإجرائية Bad_Div.

تبدأ الإجرائية Bad_Div بتوجيه للمجمّع بحيث يعرف أن اللصاقة Bad_Div_Flag تمثل متحولاً ذا ثمان خانات Byte، وأنه معرّف في قطاع خارجي يُسمى Data_Here. ثم نعرّف الإجرائية Bad_Div إجرائية عمومية Public بحيث يمكن وحداتٍ برمجيةً خارجيةً الوصولُ إليها. يلي ذلك تعريف قطاع منطقي للمعطيات نسميه IN_Proc_here. نلاحظ أنه بالإمكان كتابة إجرائية المقاطعة في القطاع المنطقي للبرنامج الرئيسي ذاته، ولكن الفصل بينهما يجعل البرنامج الكلي أكثر منهجية.

تعرّف العبارة Bad_Div Proc FAR البداية الفعلية لإجرائية المقاطعة، وتوجه المجمّع لكي يخزّن قيم السجلين CS و IP عند القفز لأن الاجراء من النوع البعيد. كما تحدد العبارة ASSUME بداية الإجرائية وتخبر المجمّع بأسماء القطاعات الواجب استخدامها للبرنامج والمعطيات.

تبدأ بعدئذٍ الإجرائية بحفظ السجلات التي ستُستخدم داخله. وينبغي تطبيق هذه العملية في جميع إجرائيات المقاطعة. تُحفظ السجلات في المكدس، لتُسترجع عند نهاية تنفيذ الإجرائية قبل تعليمة العودة IRET. وفي مثالنا، نحفظ السجلين DS و AX.

ونوجه الانتباه إلى أهمية تخزين DS، إذ من الممكن أن تحدث المقاطعة عندما يكون البرنامج في قطاع معطيات مغاير لذلك القطاع المعرّف في إجرائية المقاطعة. ولذا، يُفضل تخزين هذا السجل لأخذ الحيطة.

ثم نشحن داخل السجل DS عنوان قطاع المعطيات المستخدم داخل إجرائية المقاطعة، وبعدئذٍ نرفع راية الخطأ بكتابة القيمة 1 داخل الموقع BAD_DIV_FLAG. ونلاحظ أن الوصول إلى هذا المتحول أصبح ممكناً لأنه قد عُرف في الملف الحالي كمتحول خارجي، وفي البرنامج الرئيسي كمتحول عمومي.

تسترجع الإجرائية، بعد رفع الراية، قيمة السجلين DS و CS، ثم تعود إلى البرنامج الرئيسي بالتعليمة IRET. تختلف هذه التعليمة عن التعليمة RET (وهي تعليمة العودة من الإجرائية)، بحيث تسترجع تلقائياً سجل الرايات المخزّن في المكدس.

لنعد الآن إلى البرنامج الرئيسي. فبعد التعليمة DIV، تُفحص الراية Bad_Div_FLAG فإذا لم تكن مرفوعة، أي قيمتها معدومة، فهذا يدل على عدم حدوث المقاطعة، ومن ثَم فعملية القسمة صالحة. نخزّن إذن خارج القسمة في الموقع الذي يؤشر عليه السجل BX. أما في الحالة المعاكسة، أي عند حدوث مقاطعة، فإن المعالج يقفز إلى اللصاقة SKIP مباشرةً دون أن يخزّن خارج القسمة في جدول النتائج.

وفي كلتا الحالتين (حالة التخزين أو عدم التخزين) يمحو البرنامج راية الخطأ استعداداً لعملية القسمة التالية، ويزيد مؤشر الدخل SI بمقدار 2، ومؤشر الخرج بمقدار 1، ثم يقفز إلى اللصاقة NEXT بواسطة التعليمة LOOP، التي تفحص ذاتياً السجل CX قبل القفز.

2-7 أنواع المقاطعات في المعالج 8086

نعرض هنا بالتفصيل الطرائق المختلفة التي يتقاطع فيها المعالج 8086، وطرائق استجابة المعالج لها. وسيشمل العرض كافة أنواع المقاطعات Interrupt Types، ولذا ليس من الضروري معرفة تفاصيل كل نوع من الأنواع من القراءة الأولى للنص، ولكن يمكن العودة إلى الفقرة المناسبة حين اللزوم.

- النوع 0: القسمة على 0

يتقاطع المعالج ذاتياً بهذا النوع عندما يتعرض لعملية قسمة على القيمة 0، أو عندما يكون خارج القسمة كبيراً إلى حد يحول دون ترميزه في سجل الوجهة. ففي حالة هذا النوع، يدفع المعالج سجل الراية إلى المكدس، ويضع القيمة 0 في الرايتين IF و TF، ويدفع عنوان العودة إلى المكدس، ثم يجلب عنوان البداية لإجرائية خدمة المقاطعة من جدول متجهات المقاطعة. فيشحن في السجل CS محتوى الموقعين 02h و 03h وفي السجل IP محتوى الموقعين 00h و 01h.

ولما كانت الاستجابة لهذه المقاطعة آلية ولا يمكن إلغاؤها، فيجب توقعها عند استخدام التعليمتين DIV أو IDIV في البرنامج الرئيسي. ويمكن الحيلولة دون حدوث هذه المقاطعة بفحص المقسوم عليه والتوثق أن قيمته تختلف عن 0 قبل القسمة.

يمكن لمعالجة هذا النوع من المقاطعات كتابة إجرائية خدمة لهذه المقاطعة كما في المثال السابق. ويتميز هذا الحل بمتانته، إذ ليس من الضروري اختبار المقسوم عليه في كل مرة. فعند حدوث خطأ ما، يقفز المعالج إلى إجرائية خدمة المقاطعة للقيام بما يلزم. ويجب ألا ننسى،عند استخدام أي مقاطعة، كتابة عنوان البداية لإجرائية الخدمة في جدول متجهات المقاطعة.

- النوع 1: مقاطعة التنفيذ الخطوي

نحتاج أثناء تطوير برنامج ما إلى تنفيذه تعليمة فتعليمة، لمراقبته وتحقق صحة أدائه. إذ نفحص بعد كل تعليمة محتوى السجلات ووضع الرايات، وإذا كانت القيم سليمة نطلب منه المتابعة. وبكلمات أخرى، يتوقف البرنامج في نمط التنفيذ الخطوي بعد كل تعليمة وينتظر إيعازات جديدة من المستثمر. يسمح النوع 1 من المقاطعات Trap بتحقيق هذا التنفيذ الخطوي Single-Step Interrupt. فعند تأهيل الراية TF يتعرض المعالج آلياً لمقاطعة من النوع 1 بعد كل تعليمة ينفذها. وعند ظهور هذه المقاطعة، يدفع المعالج سجل الرايات إلى المكدس، ويضع القيمة صفر في الرايتين TF و IF، ثم يدفع محتوى السجلين CS و IP إلى المكدس لتخزين عنوان العودة. يجلب المعالج بعدئذٍ عنوان البداية لإجرائية خدمة المقاطعة من العنوانين 06h و 07h ويشحنهما في السجل CS، كما يجلب العنوانين 04h و 05h ويضعهما في السجل IP.

ويجب لاستخدام هذا النوع من المقاطعة تحقيق ما يلي:

- وضع راية التنفيذ الخطوي TFعلى 1.
- كتابة إجرائية خدمة المقاطعة المناسب، الذي يخزّن كافة السجلات المستخدمة في الإجرائية.
- شحن عنوان البداية في جدول متجهات المقاطعة بدءاً من العنوان 04h وحتى 07h.

أما محتوى الإجرائية فيختلف بحسب المهمة المطلوبة بعد التنفيذ الخطوي. فمثلاً، قد تتضمن الإجرائية إرسال قيمة السجلات إلى وحدة الإظهار ليفحصها المستثمر.

يمكّن تأهيلَ راية التنفيذ الخطوي اتباعُ التسلسل التالي:

```
PUSH F                  ; push flags on stack
MOV BP, SP              ; copy SP to BP for use as index
OR [BP + 0], 0100h      ; set TF bit
POPF                    ; restore FLAG
```

ولإلغاء هذه الراية تُستبدل بالتعليمة OR التعليمة AND [BP + 0], FEFFh.

- النوع 2: المقاطعة غير القابلة للحجب

يتعرض المعالج 8086 لمقاطعة من النوع 2، عند ظهور جبهة هابطة على مدخله NMI (ويُقصد بجبهة هابطة انتقال مستوى الإشارة المطبّقة على المدخل NMI من المستوى المرتفع إلى المستوى المنخفض). واستجابةً لهذه المقاطعة، يدفع المعالج سجل الراية إلى المكدس، ويلغي عمل الرايتين TF و IF، كما يدفع محتوى السجلين CS و IP إلى المكدس لتخزين عنوان العودة. ثم يجلب عنوان البداية لإجرائية خدمة المقاطعة من جدول متجهات المقاطعة، فيشحن في السجل IP محتوى الموقعين 08h و 09h، وفي السجل CS محتوى الموقعين 0Ah و 0Bh.

توصف هذه المقاطعة بأنها غير قابلة للحجب، لأن المعالج لايستطيع إلغاءها برمجياً. فعند ظهور الجبهة الهابطة على المدخل NMI، يقفز المعالج حتماً إلى إجرائية خدمة المقاطعة الموافقة.

تفيد هذه المقاطعة في إعلام المعالج بحدث خارجي. فقد يُوصل إلى ذلك المدخل محس الضغط لغلاية مثلاً. فإذا ارتفع الضغط فوق حد معين، فإن المحس يرسل إشارة مقاطعة إلى المعالج. وفي هذه الحالة، تستطيع إجرائية خدمة المقاطعة إعطاءَ أمر لإطفاء الغلاية، وفتح صِمام الضغط، وتشغيل الإنذار. وثمة استخدام شائع آخر لهذه المقاطعة، وهو تخزين معطيات البرنامج عند حدوث خلل في التغذية. إذ تسمح دارات خارجية بتحسس خلل التغذية، وتُرسل إشارة المقاطعة NMI إلي المعالج. ولما كان الزمن اللازم لانقطاع التيار كبيراً بالمقارنة بزمن تنفيذ التعليمات، فإن المعالج يستطيع خلال هذا الزمن، وبعد وصول المقاطعة، تخزين المعطيات المهمة في القرص الصلب، أو في أي وسيطة من وسائط التخزين الثابتة. وعند عودة

التيار، تُسترجع أولاً المعطيات المخزّنة، ويتابع المعالج العمل دون ضياع لأي معلومة.

- النوع 3: مقاطعة نقطة التوقف

تتولد مقاطعة نقطة التوقف Break Point بتنفيذ التعليمة INT 3.

إن الاستخدام الرئيسي لهذه المقاطعة هو إضافة نقطة توقف إلى البرنامج. ويُقصد بذلك توقف المعالج عن تنفيذ البرنامج عند وصوله إلى تعليمة معينة، وهذا ما يسمح بفحص محتوى سجلاته، والتوثق من صحة تنفيذ البرنامج.

وتختلف نقطة التوقف عن التنفيذ الخطوي، في أن المعالج قد ينفذ عدة تعليمات قبل الوصول إلى نقطة التوقف. أما في حالة التنفيذ الخطوي فالمعالج يتوقف حتماً بعد كل تعليمة.

لذا، عند وضع نقطة التوقف في برنامج ما تُضاف التعليمة INT 3 في تلك النقطة، فيقوم المعالج عند تنفيذ هذه التعليمة بدفع سجل الراية إلي المكدس، ويضع القيمة 0 في الرايتين TF و IF، كما يدفع السجلين CS وIP إلى المكدس لتخزين عنوان العودة. يجلب المعالج بعد ذلك عنوان البداية لإجرائية خدمة المقاطعة من جدول متجهات المقاطعة من الموقع 0Ch وحتى 0Fh. أما الإجرائية ذاتها فتخزّن كافة السجلات في المكدس. وتختلف المهمة التي تقوم بها تبعاً للنظام، فقد تقوم وحدة إظهار مثلاً بإخراج السجلات على الشاشة حتى يستطيع المستثمر تحققَ برنامجه.

- النوع 4: مقاطعة الفائض

تُرفع راية الفائض OF إذا تعذر تمثيل ناتج عملية حسابية معينة في سجل الوجهة أو في موقع من الذاكرة. فعلى سبيل المثال، إذا أضفنا العدد ثماني الخانات ذا الإشارة 0110 1100 (108_{DEC}) إلى العدد ثماني الخانات ذي الإشارة 0101 0001 (81_{DEC})، فإن النتيجة ستكون

1011 1101 (189_{DEC}). إن هذه النتيجة صحيحة إذا كان الجمع لأعداد بلا اشارة. أما في حال الأعداد ذات الاشارة، فيدل الرقم 1 في الخانة العليا على أن العدد سالب، ومرّمز بالإتمام إلى 2. ومن ثَم، تُفهم النتيجة على أنها مساوية لـ (-67_{DEC}) ، وهذا خطأ.

توجد للحيلولة دون وقوع هذه الأخطاء طريقتان رئيسيتان. الأولى تعتمد على القفز عند الفائض (تعليمة J0) وتُستخدم بعد التعليمة الحسابية مباشرة. فإذا كانت راية الفائض مرفوعة بسبب العملية الحسابية، فسيقفز المعالج إلى العنوان المحدد بالتعليمة J0؛ وفي ذلك العنوان يعالج الخطأ على الوجه المناسب. أما الطريقة الثانية فتعتمد على اكتشاف خطأ الفائض بالمقاطعة. ويجري ذلك بتعليمة INTO التي توضع مباشرة بعد التعليمة الحسابية في البرنامج. فإذا لم تُرفع راية الفائض عند تنفيذ INTO، فإن التعليمة ستتصرف كتعليمة NOP. وإلا، فإنها ستولّد مقاطعة من النوع 4. واستجابة لهذه المقاطعة، يدفع المعالج سجل الراية إلى المكدس، ويضع القيمة 0 في الرايتين TF و IF، ويدفع السجلين CS و IP إلى المكدس. ثم يجلب المعالج عنوان البداية لإجرائية خدمة المقاطعة من جدول متجهات المقاطعة من المواقع 010h وحتى 013h.

يُعالج خطأ الفائض داخل إجرائية المقاطعة، فقد توضع قيمة تدل على الخطأ في موقع ما من الذاكرة، كما فعلنا في المثال BAD_DIV. وتمتاز الطريقة الثانية عن الأولى بأنها يمكن الوصول إليها من أي مكان في البرنامج.

- المقاطعات البرمجية 0 إلى 255

يمكن أن تستخدم تعليمة INT لتوليد مقاطعة من أي نوع من الأنواع الممكنة، ويُحدد النوع في التعليمة ذاتها. فمثلاً، تؤدي التعليمة INT 32 إلى توليد مقاطعة من النوع 32. وعند حدوث مثل هذه المقاطعات البرمجية، يدفع المعالج سجل الراية إلى المكدس،

فيضع القيمة 0 في الرايتين TF و IF ويخزّن السجلين CS و IP في المكدس للحفاظ على عنوان العودة. يجلب المعالج بعد ذلك عنوان البداية لإجرائية خدمة المقاطعة بحسب النوع المحدد في التعليمة. فمثلاً، يجلب المعالج عنوان إجرائية المقاطعة من النوع 32 من الموقع 128 = 32*4 (أي 80h) و81h، ويشحنه في السجل IP. كما يضع في السجل CS محتوى العنوان 82h و 83h.

تفيد المقاطعات البرمجية في اختبار إجرائيات خدمة المقاطعة. فعلى سبيل المثال، يمكن استخدام INT 0 لتوليد مقاطعة برمجية من النوع 0 (القسمة على 0)، دون الحاجة إلى تنفيذ برنامج القسمة. كما يمكن استخدام التعليمة INT 2 لتنفيذ إجرائية المقاطعة المرتبطة بالدخل NMI دون ظهور إشارة مقاطعة خارجية.

وثمة فائدة مهمة أيضاً للمقاطعات البرمجية، وهي استدعاء الإجرائيات المطلوبة من أي برنامج. ففي الحاسوب الشخصي PC، توجد مجموعة من الإجرائيات المضمّنة في الذاكرة ROM، بحيث تؤدي كل إجرائية منها عملاً معيناً، مثل قراءة محرف من لوحة المفاتيح أو كتابة بعض المحارف على الشاشة أو قراءة محارف من القرص. تسمى مجموعة الإجرائيات هذه إجرائيات الدخل والخرج الأساسية BIOS (Basic Input Output Subroutines). تستدعى الإجرائيات BIOS بواسطة تعليمة INT. فمثلاً، إذا أردنا إرسال بعض المحارف إلى الطابعة في حاسوب شخصي، فبالإمكان استخدام إجرائية BIOS مخصصة لذلك (يستدعى بالتعليمة INT 17).

<u>مثال:</u>

```
STACK_HERE SEGMENT STACK
DW 200 DUP(0)              ; set aside 200 words for stack
STACK_TOP LABEL WORD       ; assign name to word
                           ; above stack top
STACK_HERE ENDS
CHAR_COUNT EQU 27
```

```
DATA_HERE SEGMENT
MESSAGE DB 'Hello there, How are you'
MESSAGE_END DB 0Dh, 0Ah          ; return and line feed
DATA_HERE ENDS

CODE_HERE SEGMENT
ASSUME CS: CODE_HERE, SS: STACK_HERE, DS: DATA_HERE
    MOV AX, STACK_HERE           ; initialize stack segment register
    MOV SS, AX
    MOV SP, OFFSET STACK_TOP     ; initialize stack pointer
    MOV AX, DATA_HERE            ; initialize data segment
    MOV DS, AX
    MOV AH, 01                   ; initiallize pointer port
    MOV DX, 0                    ; to use printer port 0
    INT 17h                      ; call procedure
                                 ; to initialize printer port
    LEA BX, MESSAGE              ; get to start of message
    MOV CX, CHAR_COUNT           ; set up a count variable
AGAIN:
    MOV AX, 0                    ; code to tell procedure
                                 ; to send character
    MOV AL, [BX]                 ; load character to be sent into AL
    INT 17h                      ; send character to printer
    CMP AH, 01h                  ; if character not printed
    JNE NEXT                     ; then AH=1
NOT_RDY:
    STC                          ; set carry to indicate
                                 ; message not sent
    JMP EXIT                     ; leave loop
NEXT:
    CLC                          ; clear carry flag to show
                                 ; character is sent
    INC BX                       ; address of next character
    LOOP AGAIN                   ; send the next character
EXIT:
    NOP
CODE_HERE ENDS

END
```

نلاحظ في هذا المثال أن السجلات AL, AH, DX مستخدمة لتمرير المعاملات إلى الإجرائية. ونلاحظ أيضاً أن الإجرائية مستخدمة في عمليتين مختلفتين، وهما تهيئة بوابة الطابعة، وإرسال محرف إلى الطابعة.

تُحدد العملية المطلوبة بالإجرائية بواسطة العدد الممرر في السجل AH. فإذا كان AH يساوي 1، فالهدف من الإجرائية هو تهيئة بوابة الطابعة، أما إذا كان AH يساوي 0 فالمطلوب هو إرسال المحرف المخزّن في السجل AL. وأما إذا كانت قيمة السجل AH تساوي 2، فالإجرائية المطلوبة هي قراءة حالة الطابعة وتخزينها في السجل AH. إذا لم تنجح محاولة طباعة الحرف لسبب ما، مثلاً قد تكون الطابعة غير جاهزة أو غير مهيأة أو مشغولة أو خالية من الورق، فإن العدد 01 يُعاد في السجل AH.

الميزة الرئيسية لاستدعاء الإجرائيات بهذه الطريقة هي الاستغناء عن ضرورة تخزين عنوان البداية للإجرائية. فكل ما يجب معرفته هو نوع المقاطعة الواجب استخدامها ودلالة السجلات اللازمة لتمرير المعاملات.

- مقاطعات الكيان الصلب 0 إلى 255

يسمح مدخل المعالج INTR لإشارات خارجية أن تقاطع عمل المعالج. وخلافاً للمدخل NMI، فإن المقاطعات على هذا المدخل يمكن إلغاؤها (أو حجبها) برمجياً. فإذا كانت راية المقاطعة IF تساوي 0، حُجبت المقاطعات على المدخل INTR. و يمكن محو راية المقاطعة IF بواسطة التعليمة CLI. و يؤهل المدخل INTR من جديد عند كتابة القيمة 1في الراية IF بواسطة التعليمة STI.

عند إقلاع المعالج 8086 تمحى راية المقاطعة IF آلياً. ولذا، ينبغي كتابة القيمة 1 في الراية IF لكي يستطيع المعالج استقبال المقاطعات على المدخل INTR. ويكمن السبب وراء المحو الذاتي عند الإقلاع، في

السماح للمعالج باستهلال الطرفيات الموصولة به قبل أن يصبح جاهزاً لاستقبال المقاطعات منها. فالمرء مثلاً يحتسي القهوة الصباحية قبل أن يصل الهاتف و يصبح مستعداً لاستقبال المقاطعات منه!

تُمحى راية المقاطعة ذاتياً أيضاً عندما يستجيب المعالج لأي مقاطعة أخرى. و يحدث ذلك لسببين:

- منع أي إشارة على المدخل INTR من مقاطعة إجرائية خدمة لمقاطعة أكثر أولوية. ومع ذلك يمكن تأهيل راية المقاطعة IF داخل إجرائية الخدمة لتأهيل المدخل INTR والسماح لهذه المقاطعات بأن توقف إجرائية الخدمة الحالية.

- الحيلولة دون حدوث عدد غير منتهٍ من المقاطعات بسبب المدخل INTR. فهذا المدخل فعّال على المستوى المرتفع للإشارة. فإذا وُجدت مثل هذه الإشارة وكانت راية المقاطعة IF مرفوعة فإن المعالج يقفز إلى إجرائية المقاطعة الموافقة. فإذا لم تحجب المقاطعة ذاتياً مع القفز إلى إجرائية الخدمة، يُقاطع المعالج ثانيةً وثالثةً مادامت الإشارة ذات المستوى المرتفع موجودة على المدخل INTR.

تؤدي التعليمة IRET في نهاية الإجرائية إلى استرجاع سجل الرايات، وتأهيل مدخل المقاطعة INTR من جديد. فإذا كان مستوى الإشارة مرتفعاً يُقاطع المعالج من جديد.

تختلف استجابة المعالج للمقاطعة INTR بعض الشيء عن باقي المقاطعات. ويكمن الاختلاف الرئيسي في أن تحديد نوع المقاطعة الذي تحدده وحدة طرفية خارجية بدلاً من أن يكون محدداً سلفاً.

يقوم المعالج أولاً بإشعار الطرفيات بقبوله للمقاطعة INTR بواسطة الخرج INTA. وهدف هذا الإشعار، الحصولُ على نوع المقاطعة المطلوب تنفيذها. وعندما يستقبل المعالج نوع المقاطعة، يدفع سجل الرايات إلى المكدس، ويمسح الرايتين IF و TF، كما يدفع محتوى السجلين CS و IP إلى المكدس لتخزين عنوان العودة. ثم يستخدم بعد ذلك قيمة النوع المقروءة من الوحدة الطرفية الخارجية لجلب عنوان

البداية لإجرائية خدمة المقاطعة. فيشحن في السجل IP محتوى الذاكرة ذات العنوان: 4× (نوع المقاطعة)، ويضع في السجل CS محتوى الذاكرة ذات العنوان: 4× (نوع المقاطعة) + 2.

يفيد تحديد نوع المقاطعة بطرفية خارجية في إمكان وصل عدة طرفيات معاً على خط المقاطعة ذاته INTR. وتختلف هذه الطرفيات بعضها عن بعض، بنوع المقاطعة التي ترسلها. فتُعطى كل طرفية رقماً مختلفاً عن غيرها، تستخدمه في تحديد نوع المقاطعة، ويُكتب لكل طرفية إجرائية خدمة تتناسب مع النوع المحدد لها.

ملاحظة: أولويات المقاطعة في المعالج 8086

السؤال الذي يطرح ذاته هو: ماذا يحدث إذا ظهرت مقاطعتان معاً في الوقت نفسه؟ والجواب أن المقاطعة ذات الأولوية العليا تُخدم أولاً. وبعد انتهاء خدمتها، ينتقل المعالج إلى المقاطعة التي تليها في سلّم الأولوية فيخدمها، وهكذا...

يظهر الشكل 15 أولويات المقاطعات في المعالج 8086.

المقاطعة	الأولوية
القسمة على الصفر INT n INTO	العليا
NMI	
INTR	
التنفيذ الخطوي	الدنيا

الشكل 15: أولويات المقاطعة في المعالج 8086.

وكمثال أول لنفترض أن المدخل INTR مؤهل، فيستقبل المعالج المقاطعة INTR أثناء تنفيذه لعملية القسمة. ولنفترض أن عملية القسمة قد ولّدت مقاطعة من النوع 0 (القسمة على 0). ولما كانت المقاطعة من النوع 0 ذات أولوية أعلى من المقاطعة INTR، فإن المعالج يخدم أولاً المقاطعة من النوع 0. وفي بداية الإجرائية، يحجب المعالج المقاطعات بمسحه للرايتين IF و TF. وبعد تنفيذ التعليمة IRET في نهاية إجرائية المقاطعة من النوع 0، يسترجع قيم الرايات كما كانت قبل حدوث مقاطعة النوع 0. وهذا ما يؤهل ثانيةً المدخل INTR، ومن ثَم يقفز المعالج إلى إجرائية المقاطعة INTR.

يظهر تسلسل مشابه للعمليات إذا كان المعالج 8086 ينفذ إجرائية المقاطعة INT 0 أو INT n عند ظهور المقاطعة INTR.

لنفترض في المثال الثاني ظهور إشارة ذات جبهة صاعدة (أي إن الإشارة انتقلت من المستوى المنخفض إلى المستوى المرتفع) على مدخل المعالج NMI أثناء تنفيذه للتعليمة DIV. ولنفترض أن هذه التعليمة قد ولّدت مقاطعة من النوع 0 (القسمة على 0). ولما كان المعالج يفحص مقاطعاته داخلياً قبل فحص المقاطعة NMI، فإنه يدفع إلى المكدس بسجل الرايات، ويمسح الرايتين TF وIF، ويدفع عنوان العودة إلى المكدس، ويقفز إلى بداية إجرائية المقاطعة من النوع 0. ولما كان المدخل NMI لا يُحجب برمجياً، فإن المعالج يستجيب للمقاطعة NMI. وبمعنى آخر، فإن المعالج يدفع الرايات إلى المكدس، ويمسح الرايتين TF و IF، ويخزن عنوان العودة، ثم يقفز إلى عنوان البداية لإجرائية المقاطعة NMI. وبعد أن ينتهي المعالج من تلك الإجرائية، يعود إلى إجرائية المقاطعة من النوع 0، فيتمّه ويعود بعد ذلك إلى البرنامج الرئيسي.

لنأخذ الآن حالة مقاطعة التنفيذ الخطوي. فإذا رُفعت راية التنفيذ الخطوي TF يتعرض المعالج لمقاطعة من النوع 1 بعد تنفيذ كل تعليمة. ولكن أثناء استجابته لهذه المقاطعة، يمسح الراية TF، وهذا ما يحول دون توليد هذه المقاطعة أثناء تنفيذ إجرائية الخدمة.

ومن جهة أخرى، إذا تقاطع المعالج بمقاطعة تختلف عن التنفيذ الخطوي وكانت راية التنفيذ الخطوي مرفوعة، فإنه يقفز إلى إجرائية خدمة المقاطعة من النوع 1 قبل أن يقفز إلى إجرائية خدمة المقاطعة الأخرى. ذلك أن أولوية التنفيذ الخطوي أعلى من بقية المقاطعات. وإذا أردنا تتبع المعالج خطوياً أثناء تنفيذه لإجرائية المقاطعة، وجب تأهيل هذه الراية (TF) داخل إجرائية المقاطعة.

الفصل الخامس

الحواسيب التفرعية

1 مقدمة

تتطلب بعض التطبيقات المعلوماتية إنجاز كميات كبيرة من العمليات الحسابية في وقت محدد، مثل التطبيقات التي تتعامل مع الصور الرقمية أو الصوت الرقمي. ففي تلك التطبيقات يمكن اتباع أحد حلين لتحقيق المتطلبات الحسابية:

- إما استخدام معالج ذي تردد عمل مرتفع وذي بنية متطورة لكي يستطيع إنجاز الكم الهائل من العمليات في الوقت المحدد. ولكن هذه الأسلوب غير متاح دوماً، لأن العمل المطلوب من المعالج، وإن ازدادت سرعته، قد يبقى أكبر من طاقته.
- أو استخدام أكثر من معالج في الحاسوب ذاته للقيام بالمهمات المطلوبة. ويُقسم العمل بين هذه المعالجات، التي تقوم بأداء المطلوب منها على التوازي. ومن هنا سميت هذه البنى الحواسيب المتوازية أو التفرعية Parallel Computers.

نعطي في هذا الفصل لمحة إلى الحوسبة التفرعية، فنصف أولاً أهم المبادئ المستخدمة في الحواسيب التفرعية، ثم نعرض التصنيف المعتمد لهذه الحواسيب.

2 مبادئ الحواسيب التفرعية

1-2 قانون أمدال

وضع العالم أمدال AMDAHL قانوناً في عام 1967 يسمح بحساب معدل التسريع الناتج من استخدام عدة معالجات بدلاً من معالج واحد. فإذا فرضنا أن الزمن اللازم لتنفيذ البرنامج بواسطة معالج وحيد تنفيذاً كاملاً هو T، وإذا وُزع الجزء f من البرنامج (حيث $0< f < 1$) على عدد لانهائي من المعالجات، فإن زمن تنفيذ البرنامج الكلي سيصبح: T(1-f). ومن ثَم يكون معدل التسريع الحاصل R هو:

$$R = \frac{T}{T(1-f)} = \frac{1}{1-f}$$

نستنتج مما سبق، أنه في الحالة المثلى لا نستطيع أن نقلّص زمن التنفيذ الكلي إلى أكثر من $\frac{1}{1-f}$. ونقترب من هذه النسبة كلما ازداد عدد المعالجات العاملة بالتوازي.

مثال:

إذا افترضنا أن نصف البرنامج قد وُزع على أربعة معالجات، فما هو معدل التسريع الأمثل الممكن الحصول عليه؟

بمقتضى قانون أمدال، فإن معدل التسريع في حال وجود عدد لانهائي من المعالجات هو:

$$R = \frac{1}{1-f} = \frac{1}{1-0.5} = 2$$

ولكن عدد المعالجات المستخدمة في مثالنا محدود (وهو أربعة)، لذا فالزمن الأمثل لتنفيذ الجزء الموزع على N معالج هو $T_{exe} = \frac{f.T}{N}$ حيث f.T زمن التنفيذ الأمثل.

وفي مثالنا نجد:

$$T_{exe} = \frac{0.5\,T}{4} = \frac{1}{8}\,T$$

ويكون الزمن اللازم لتنفيذ البرنامج الكامل هو:

$$T(1-\frac{1}{8}) = \frac{7}{8}T$$

ومن ثَم يصبح معدل التسريع R الممكن الوصول إليه:

$$R = \frac{T}{\frac{7}{8}T} = \frac{8}{7} = 1.14$$

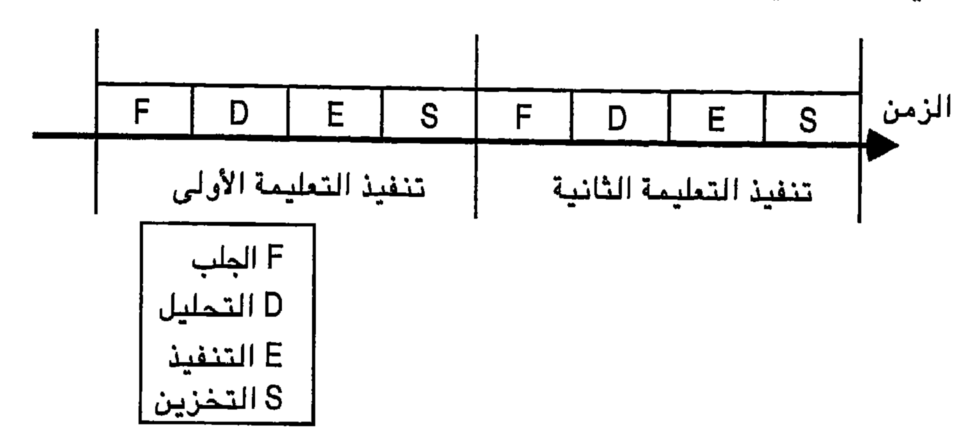
وهكذا نجد أن معدل التسريع R يبتعد عن تلك القيمة المعطاة بقانون أمدال، وذلك بسبب استخدام عدد محدود (منته) من المعالجات. وتُعدّ هذه القيمة حداً أعلى لا يمكن تحقيقه، لأنه في التطبيق الفعلي تحتاج المعالجات إلى تمرير بعض المعطيات الثانوية فيما بينها لإنجاز المهمات المطلوبة منها. ومن ثَم، فإن معدل التسريع الفعلي يكون غالباً أدنى من المعدل الأمثل.

2-2 مبدأ المعالجة التواردية

ذكرنا سابقاً أن المراحل الأساسية لتنفيذ تعليمة ما هي:

- مرحلة الجلب Fetch: وفيها يقرأ المعالج رمز التعليمة من الذاكرة.
- مرحلة فك الترميز Decoding: يقوم المعالج بتحليل الرمز المقروء لمعرفة ما يجب أداؤه.
- مرحلة التنفيذ Execution: بعد تحليل التعليمة، قد يضطر المعالج إلى جلب بعض المعاملات من الذاكرة. إن هذه العملية ضرورية لتنفيذ التعليمة. فمثلاً، يتطلب تنفيذ التعليمة MOV A, [SI] قراءة محتوى الذاكرة المؤشر عليها بالسجل SI.
- مرحلة التخزين Storage: قد تتطلب بعض التعليمات تخزين النتيجة النهائية في الذاكرة. ويجري ذلك في هذه المرحلة.

ينفذ المعالج «التقليدي» تعليمات البرنامج واحدة تلو الأخرى تنفيذاً تسلسلياً. ويبين الشكل 1 تمثيلاً لمراحل التنفيذ التسلسلي.



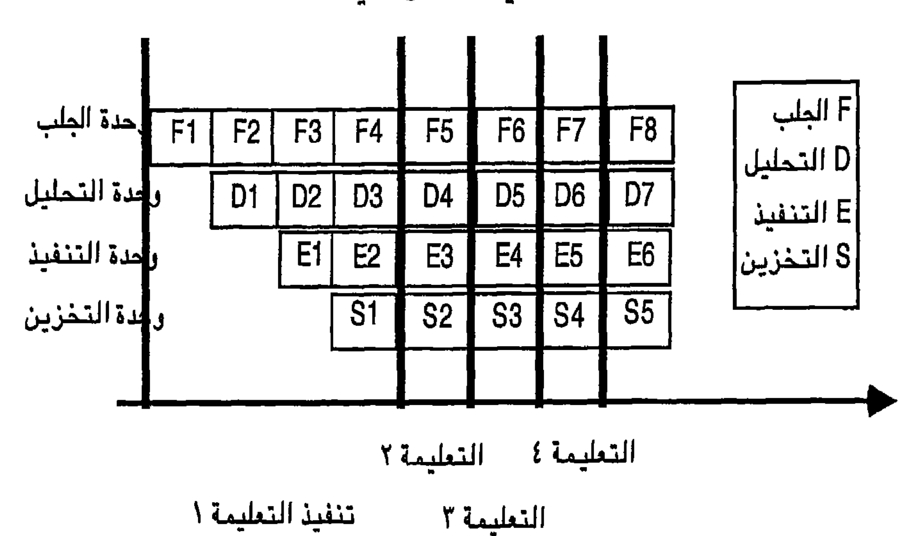
الشكل 1: مراحل التنفيذ التسلسلي في المعالجات التقليدية.

وعلى سبيل المماثلة، فإن عمل هذا المعالج يشبه دور العامل في مصنع السيارات الذي كُلف بأداء كل عمليات الإنتاج. إذ يجب عليه أن يصنع الهيكل المعدني، فإذا فرغ منه وضع المحرك فيه، وبعد ذلك، يضع المقاعد وأخيراً، يُلصِق بطاقة تحدد سعر السيارة!

ومن البديهي، أن تسريع العمل في المعمل يقتضي توظيف عدة عمال على خط الإنتاج، يختص كل منهم بجزء من العملية الإنتاجية. فمثلاً، يختص الأول بصناعة الهيكل المعدني، والثاني بتركيب المحرك، والثالث بوضع المقاعد، والأخير بإلصاق سعر السيارة.

إن المبدأ ذاته يمكن تطبيقه في حالة المعالجات. فبدلاً من إجراء المراحل الأربع تسلسلياً، يمكن تنفيذها دفعة واحدة إذا توفرت لكل مرحلة وحدة تخصصية مناسبة. يسمى ذلك المبدأ التوارد Pipelining. وقد سمي بذلك، لأن كل وحدة تخصصية تورّد نتائجها إلى الوحدة التي تليها عند إنهاء مهمتها عبر «قناة» نسميها قناة الموارِدة Pipeline.

وقد يوجد في بعض المعالجات المتقدمة وحدة متخصصة لجلب التعليمات، وأخرى متخصصة بتحليلها، ووحدة ثالثة متخصصة بتنفيذها، وأخرى متخصصة في تخزين النتائج. وعندئذٍ تُنفذ التعليمات وفق المخطط الزمني الموضح في الشكل 2.



الشكل 2: التنفيذ وفق مبدأ التوارد.

نجد من المخطط السابق، أن التعليمة الأولى قد احتاجت إلى أربعة أدوار للتنفيذ – مثلها كمثل أي تعليمة في المعالج التقليدي، في حين استغرق تنفيذ التعليمة 4 دوراً واحداً فقط، وكذلك التعليمات التالية لها. نستنتج إذن أن وجود الوحدات التخصصية الأربع قد قلّص زمن التنفيذ إلى الربع تقريباً، ما عدا التعليمة الأولى التي يبقى زمن تنفيذها دون تغيير، وتسمى اللحظة التي ينتهي فيها تنفيذ التعليمة الأولى لحظة امتلاء القناة، وذلك لأن لكل وحدة تخصصية نتيجة جاهزة بدءاً من تلك اللحظة.

ولقد ذكرنا أن نسبة التسريع الناتجة هي -على وجه التقريب- مساوية لأربعة. ويكمن السبب في كونها تقريبية في أن بعض

التعليمات «تحطم» التنفيذ التسلسلي للبرنامج مثل تعليمات القفز. فهذه التعليمات تؤدي إلى إفراغ القناة، ثم ملئها مجدداً بعد إنجاز عملية القفز، وهذا ما يبطئ سرعة تنفيذ البرنامج.

3-2 تطبيق التوارد على المعطيات

ذكرنا سابقاً أن للذاكرة زمناً يحدد سرعة استجابتها ويسمى زمن النفاذ Access Time. إن هذا الزمن يفصل بين لحظة طلب المعلومة من الذاكرة ولحظة الحصول عليها، في حالة القراءة منها؛ أو بين لحظة تقديم المعلومة المراد كتابتها ولحظة انتهاء الكتابة، في حالة الكتابة فيها.

جعل التقدم الذي شهدته تقانة المعالجات الصغرية زمن النفاذ إلى الذاكرة عقبة يجب تجاوزها، وذلك لتسريع التخاطب معها. وهذا ما يحول دون الاستفادة من كامل أداء المعالجات.

لقد سمحت تقانة تصنيع الذواكر بتقليص زمن النفاذ، ولكنه مع ذلك بقي دون متطلبات المعالجة السريعة. ولذا تَوجّه البحث نحو تقنية جديدة في تبادل المعطيات مع الذاكرة، وهي توارد المعطيات Data Pipeline.

تنص هذه التقنية على وضع عناوين المواقع المراد استخدامها في قناة الدخل، وتقوم الذاكرة بقراءة هذه العناوين ووضع المعطيات في قناة الخرج. تحتوي هذه الذاكرة إذن على وحدتين: تهتم الأولى بتحصيل العناوين من المعالج، و تقوم الثانية بوضع المعطيات في الخرج كما هو موضح في الشكل 3.

وحدة الدخل	العنوان ١	العنوان ٢	العنوان ٣	العنوان ٤	العنوان ٥
وحدة الخرج			المعطيات ١	المعطيات ٢	المعطيات ٣

تأخير دور واحد زمن الوصول إلى الذاكرة

الشكل 3: توارد المعطيات في الذاكرة.

4-2 البنيان السُلّمي الفائق

كما ذكرنا آنفاً، تزداد سرعة تنفيذ البرنامج عند تطبيق مبدأ التوارد. ووفقاً لذلك المبدأ، يقوم المعالج بإنجاز مراحل تنفيذ التعليمة الأربع تنفيذاً متوازياً، ولكنه لا ينفّذ إلا تعليمة واحدة في وقت واحد.

لتحسين الأداء، عمدت بعض الشركات إلى تصنيع معالجات عالية الأداء تستطيع جلب عدة تعليمات في وقت واحد وتحليلها وتنفيذها وتخزين النتائج. وبمعنى آخر، ينفّذ المعالج عدة تعليمات في آنٍ واحد. توصف هذه المعالجات بأنها معالجات سلّمية فائقة Superscalar. يوضح الشكل 4 آلية تنفيذ التعليمات في تلك المعالجات.

الشكل 4: تنفيذ التعليمات في المعالجات السلّمية الفائقة.

نلاحظ في الشكل 4، كما هو حال التوارد، أن تنفيذ أول دفعة من التعليمات تتطلب زمناً قدره أربعة أدوار، في حين لا تحتاج الدفعات التالية إلا إلى دور واحد.

يملك المعالج السلّمي الفائق عدة وحدات حسابية ومنطقية ALU تعمل على التوازي، وهذه الوحدات تُبرمج بتعليمات مرمّزة على عدد

كبير من الخانات (مثلاً 96 bits)، وتسمى بالتعليمات ذات الكلمات العريضة جداً VLIW (Very Large Instruction Word).

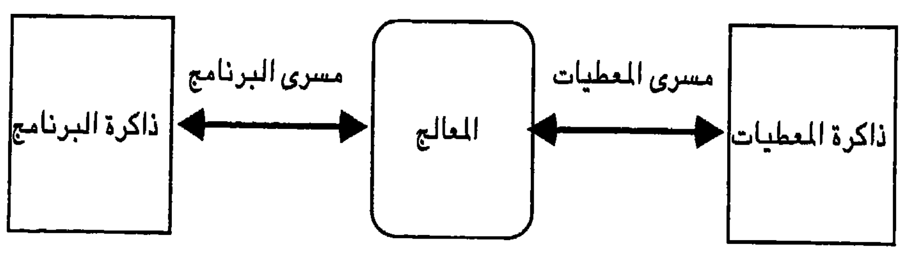
وكمثال على هذا البنيان نأخذ المعالج Pentium الذي يملك وحدتي حساب للأعداد الصحيحة Integers، ويمكنهما العمل معاً إذا توافرت الشروط الملائمة لذلك.

5-2 بنيان هارفرد

رأينا سابقاً أن المعالج يستعمل الذاكرة لتخزين المعطيات والبرنامج معاً. فهو إذن يتصل بالذاكرة بواسطة مسرى وحيد. إن لهذا البنيان أثراً مباشراً في أداء الحاسوب. فالمعالج لا يستطيع قراءة تعليمة من الذاكرة وكتابة نتيجة عملية سابقة في آنٍ واحد، وسبب ذلك استخدامه للمسرى ذاته للنفاذ إلى الذاكرة.

لتحسين الأداء، اعتمدت بعض الشركات المصنعة بنياناً ذا مسريين مستقلين:

- المسرى الأول يتصل بذاكرة البرنامج، و يناط به نقل التعليمات من الذاكرة إلى المعالج.
- المسرى الثاني يتصل بذاكرة المعطيات، ويسمح بتبادل المعطيات بين الذاكرة والمعالج. يسمى هذا البنيان بنيان هارفرد، وهو موضح في الشكل 5.



الشكل 5: بنيان هارفرد.

تتمتع المعالجات ذات بنيان هارفرد بالقدرة على النفاذ إلى الذاكرتين في آنٍ واحد. ومن ثَم، فجلب التعليمات في تلك المعالجات لا يعوق تبادل المعطيات مع الذاكرة، وهذا من شأنه أن يرفع أداء الحاسوب.

ونذكر كمثال على هذا البنيان معالجات الإشارة الرقمية DSP (Digital Signal Processor)، وهي معالجات صغرية لها بنية موجَّهة لتنفيذ الخوارزميات المصادَفة في تطبيقات الصوت الرقمي والصورة الرقمية، وما شابههما...

تكمن المثلبة الرئيسية لهذا البنيان في عدد المرابط الكبير. فللمعالج مسريان بدلاً من مسرى واحد، وهذا ما يزيد من حجمه ومن عدد مرابطه، ومن ثَم يصبح توصيله داخل النظام معقداً.

3 تصنيف البنى التفرعية

إن التنوع الكبير في الحواسيب التفرعية ومبادئها، وطرق الاتصال البينية، يجعل تمييزها أمراً شاقاً. وفي الواقع، وُضعت معايير كثيرة لتصنيف الحواسيب التفرعية في مجموعات، بحيث يكون لكل مجموعة سماتها المميزة، وملامحها الخاصة بها. ولكن إيجاد مثل هذا التصنيف الشمولي الذي ينطبق على جميع الحواسيب التفرعية أمر محال، إذ يمكن دوماً إيجاد بنى تفرعية تنتمي إلى أكثر من مجموعة في آنٍ واحد، كما يمكن إيجاد بنى تفرعية أخرى لا يمكن تصنيفها في أي مجموعة.

تخرج مناقشة نقاط الاختلاف بين معايير التصنيف عن مضمار هذا الفصل. لذا سنكتفي هنا بذكر التصنيف الأكثر شيوعاً، والذي اقترحه العالم فلين Flynn في عام 1967.

صنف فلين الحواسيب التفرعية وفق معيارين هما:

- تدفق التعليمات؛
- تدفق المعطيات.

فقد تكون التعليمات موحَّدة بين العقد الحسابية أو مختلفة فيما بينها، وكذلك قد تكون المعطيات موحَّدة أو مختلفة. وبذا، نحصل على أربع فئات ممكنة هي:

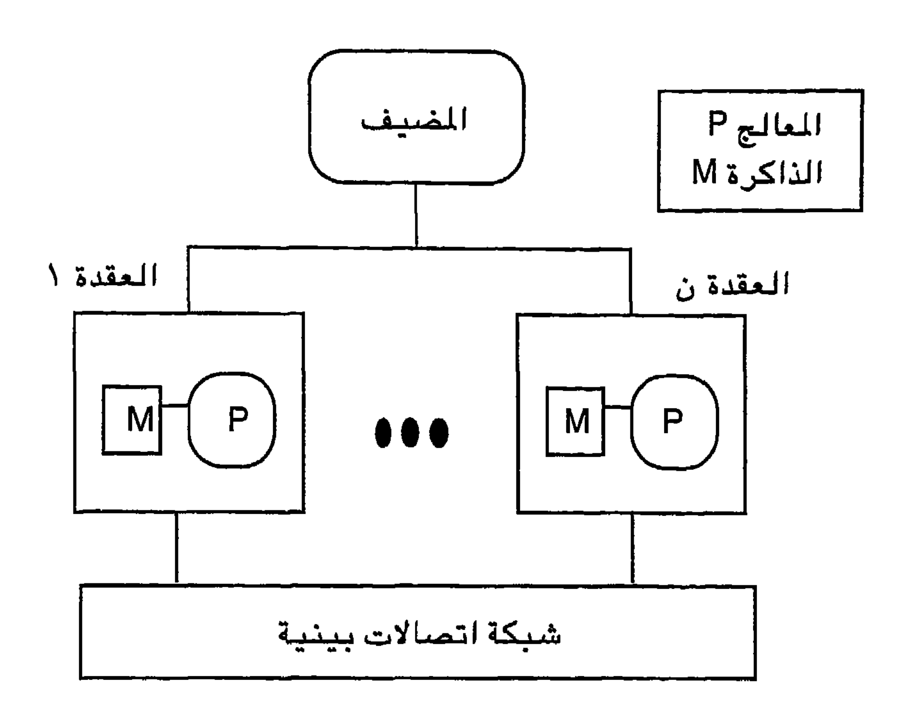
- البنية ذات التعليمات الموحدة والمعطيات الموحدة SISD (Single Instruction - Single Data)؛
- البنية ذات التعليمات الموحدة والمعطيات المختلفة SIMD (Single Instruction - Multiple Data)؛
- البنية ذات التعليمات المختلفة والمعطيات الموحدة MISD (Multiple Instruction - Single Data)؛
- البنية ذات التعليمات المختلفة والمعطيات المختلفة MIMD (Multiple Instruction - Multiple Data). وسنعرض هذه البنى تباعاً.

1-3 البنية ذات التعليمات الموحدة والمعطيات الموحدة

وهي تمثل بنية المعالج التقليدي (الذي ينطبق عليه قانون فون نويمان)، إذ يملك هذا المعالج ممراً وحيداً للتعليمات، وممراً وحيداً للمعطيات.

2-3 البنية ذات التعليمات الموحدة والمعطيات المختلفة

في هذه البنية، تُوزع التعليمة المراد تنفيذها على جميع العقد، وتقوم هذه العقد بتنفيذ التعليمة ذاتها ولكن على معطيات مختلفة. فإذا أردنا، مثلاً، تنفيذ عملية ضرب لـ N رقماً بالقيمة 2، فتُوزع أولاً تعليمة الضرب على العقد، ثم تُوزع الأرقام (N) على العقد، ثم تقوم كل عقدة بإجراء عملية الضرب داخلياً. وهكذا، تحتاج هذه البنية إلى زمن تنفيذ عملية ضرب واحدة لإجراء N عملية ضرب. تمثل هذه البنية بالمخطط الموضح في الشكل 6.



الشكل 6: البنية التفرعية SIMD.

3-3 البنية ذات التعليمات المختلفة والمعطيات الموحدة

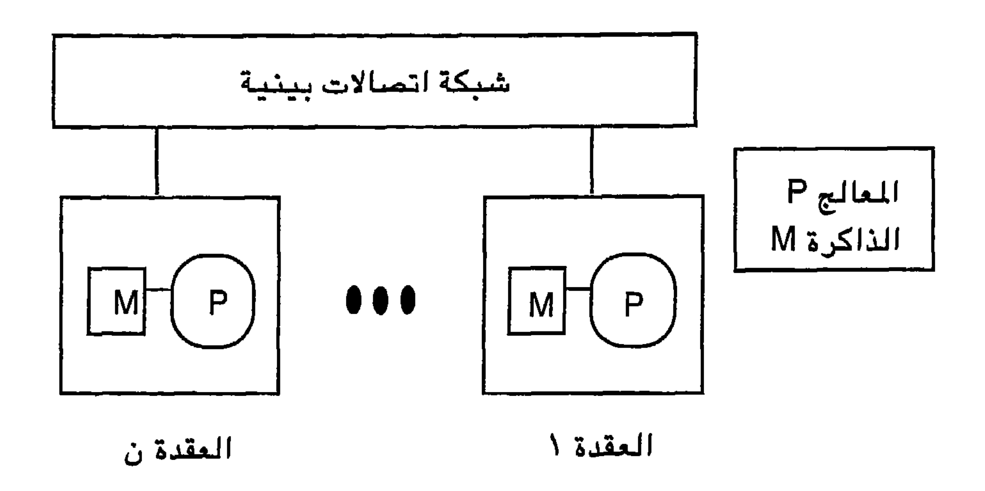
تنفذ العقد الحسابية، في هذه البنية، تعليمات مختلفٌ بعضُها عن بعض، على المعطيات ذاتها. وتندر عملياً التطبيقات التي نحتاج فيها إلى هذه البنى، وهذا ما يفسر قلة الحواسيب التفرعية المصمّمة وفق هذه البنية.

4-3 البنية ذات التعليمات المختلفة والمعطيات المختلفة

تتألف هذه البنية من مجموعة من العقد الحسابية التي تقوم بتنفيذ تعليمات مختلفة على معطيات متعددة. وهذه البنية هي أكثر البنى كفاءة، إذ يقوم الحاسوب المركزي (والذي يسمى المضيف Host)

في البداية بتوزيع المهمات بين العقد، ثم يقوم بتوزيع المعطيات على هذه العقد، وينتقل بعد ذلك إلى دور المراقبة.

تقوم عندئذٍ العقد بتنفيذ مهماتها تنفيذاً منفصلاً عن الحاسوب المضيف، وتعمل مستقلةً بعضها عن بعض. يظهر الشكل 7 مخططاً لهذه البنية.



الشكل 7: البنية التفرعية MIMD.

إن ظهور حواسيب تفرعية حديثة جعل هذا التصنيف غير كافٍ. فمثلاً، لا يمكن ضم البنية المتواردة إلى صف البنى SIMD، لأن مراحل القناة المختلفة لا تنفذ التعليمة ذاتها، كما لا يمكن عدّها مع البنى MIMD لأن العقد لا تعمل مستقلةً بعضها عن بعض.

4 أنواع المعالجة التفرعية

نشير آخراً إلى ضرورة التمييز بين نوعين من المعالجة التفرعية:

- المعالجة التفرعية على المعطيات Data Parallel Processing، وفيها

تُنفذ مجموعة من المهمات المتعاقبة على المعطيات الموزعة بين العقد الحسابية، والمخزنة في الذواكر المحلية لهذه العقد. بمعنى آخر، عندما تكون كمية المعطيات كبيرة، فمن الأفضل توزيعها على عدة عقد ثم تنفيذ الخطوات الحسابية ذاتها على كل جزء منها. ينطبق هذا النوع من المعالجة على البنى المشابهة لـ SIMD.

• المعالجة التفرعية على المهمات Task Parallel Processing، في هذا النوع، تسمح المعالجة التفرعية بتجزئة تطبيق معين إلى مهمات مستقلة، قابلة للتنفيذ على التوازي. أما المعطيات فيجب أن تكون مخزنة في ذاكرة، بحيث يمكن لجميع المهمات الوصول إليها. بمعنى آخر، يجب تخزين المعطيات في ذاكرة مشتركة.

أما البرنامج فإنه يتألف من مجموعة من المهمات «المتسايرة». ينطبق هذا النوع من المعالجة على البنى MIMD التي تسمح للعقد أن يكون لها برنامجها المستقل عن غيرها، على أن تستطيع كل عقدة الوصول إلى المعطيات اللازمة لأداء مهمتها.

الفصل السادس

بنيان الحواسيب

ذات مجموعة التعليمات الموجزة

1 مقدمة

درسنا في فصول سابقة من هذا الكتاب بنية المعالجات الصغرية «الاعتيادية» وأخذنا مثالاً عليها المعالج 8086 وعائلته. تنتمي هذه المعالجات إلى فئة المعالجات ذات «مجموعة التعليمات الموسَّعة [1]» CISC؛ وقد أطلقت عليها هذه التسمية لأن كل معالج (أو حاسوب [2]) فيها روعي عند تصميمه أخذ كافة الوظائف التي يستطيع «أسلافه» تحقيقها، ثم أضيفت إليه وظائف جديدة. مثال ذلك الزيادات والتحسينات في المقدرة والأداء التي يتمتع بها المعالج 80286 بالمقارنة بالـ 8086، والمعالج 80386 بالمقارنة بالـ80286، وهكذا...

وجد هذا النهج «التصاعدي» مبرراته في محاولة المصممين تقليص عرض الهوة بين لغات الآلة ولغات البرمجة العالية المستوى (مثل C أو Pascal) عن طريق زيادة مقدرة لغة الآلة إلى أقصى حد تسمح به قيود التقانة. نتيجة لذلك، ظهرت إلى حيز الوجود معالجات

1 أو "المعقدة" إذا أردنا الترجمة الحرفية للكلمة!

2 سنستخدم في هذا الفصل كلمتي "حاسوب" و "معالج" كمترادفين.

ذات مجموعة تعليمات عالية التعقيد، تتميز (1) بكثرة عدد التعليمات بأنواعها المختلفة (التعليمات الحسابية، تعليمات النفاذ إلى الذاكرة، الخ...)؛ و (2) بزيادة عدد أنماط العنونة زيادة ملحوظة؛ و (3) بتمكنها من التعامل مع كلمات متزايدة العرض: 8 خانات اثنانية ← 16 خانة ← 32 خانة ← 64 خانة... لقد ظن المصممون من أنصار هذا النهج أن أداء المعالج يتعلق أساساً بمقدرته على تنفيذ تعليمات معقدة؛ إذ رأوا أنهم بزيادة تعقيد مجموعة تعليمات المعالج يقللون حتماً من عدد التعليمات الناتجة من ترجمة برنامج مكتوب بلغة عالية المستوى إلى لغة الآلة.

غير أن بعض الباحثين أخذوا، منذ أوائل الثمانينيات تقريباً، يشككون بتلك «المسلّمة» التي بدت بديهية لأنصار مجموعة التعليمات الموسعة. وبدأ هؤلاء يدرسون إحصائياً نسب استخدام التعليمات المختلفة في مجموعة تعليمات المعالجات. إحدى أشهر هذه الدراسات تلك التي قام بها فيركلو Fairclough على المعالج 68000 (من صنع شركة Motorola) وهو معالج من نمط CISC يماثل تقريباً في قدرته المعالج 8086.

قسم فيركلو مجموعة تعليمات الـ 68000 إلى ثمانية أصناف هي:

1 تعليمات نقل المعطيات من موضع إلى آخر؛
2 تعليمات التحكم في تسلسل تنفيذ البرنامج؛
3 العمليات الحسابية؛
4 تعليمات المقارنة؛
5 العمليات المنطقية؛
6 تعليمات الإزاحة؛
7 تعليمات التعامل مع الخانات الاثنانية المنفردة؛
8 تعليمات الدخل/الخرج + ما تبقى من تعليمات.

ثم قام بدراسة معدل ورود كل صنف من تلك الأصناف إحصائياً في مجموعة كبيرة من البرامج المكتوبة بلغات عالية المستوى (C, Pascal, Fortan, ...) والمترجمة إلى لغة الآلة 68000. وقد اختار

مجموعة البرامج تلك بحيث تُمَثِّل غالب التطبيقات البرمجية المألوفة، فوجد النتائج المعروضة في الجدول 1.

صنف التعليمة	نسبة ورود الصنف في مجموعة البرامج المفحوصة
1	43.52%
2	25.13%
3	12.09%
4	9.15%
5	5.03%
6	2.65%
7	2.36%
8	0.07%
	100.00%

الجدول 1

نلاحظ في هذا الجدول أن صنف تعليمات نقل المعطيات يمثل بمفرده 43.52% من مجموع التعليمات المستخدمة عملياً، وأن نسبة استخدام العمليات الحسابية والمنطقية وتعليمات المقارنة والإزاحة (الأصناف 3، 4، 5، 6) لاتكاد تصل إلى 30% من مجموع التعليمات المستخدمة عملياً.

تابع فيركلو دراسته الإحصائية على مجموعة تعليمات المعالج 68000، فدرس نسب استخدام التعليمات واحدة واحدة، ووجد النتائج المعروضة في الجدول 2.

وفقاً لهذا الجدول، فإن 30.3% من مجمل مجموعة تعليمات المعالج المذكور لم تُستخدم إطلاقاً في كافة البرامج المفحوصة! أما نسبة التعليمات التي تتميز بمعدل استخدام يفوق الـ 5.0% فلا تتجاوز 2.6% من مجمل مجموعة التعليمات! وقد استنتج فركلو نتيجة لذلك أن اختصار 31 تعليمة من مجمل مجموعة تعليمات الـ 68000، البالغ عددها 76، لن يكون له أثر كبير في أداء المعالج!

النسبة إلى مجموع التعليمات المتاحة في الـ 68000	معدل الورود في مجمل البرامج المفحوصة
30.3%	= 0.0%
3.9%	≤ 0.1%
21.1%	≤ 0.5%
11.8%	≤ 1.5%
10.5%	≤ 2.0%
13.2%	≤ 3.0%
0.0%	≤ 4.0%
6.6%	≤ 5.0%
2.6%	> 5.0%
100.0%	

الجدول 2

دفعت هذه الدراسة (ودراسات أخرى عديدة أعطت نتائج متوافقة) الباحثين في بنية الحواسيب والمعالجات الصغرية إلى التساؤل حول مغزى الزيادة المطّردة في تعقيد مجموعة التعليمات، تلك الزيادة التي تؤدي وضوحاً إلى زيادة تعقيد بنية المعالج القادر على تنفيذها زيادة لا مبرر لها، خاصة وأن تعقيد البنية قد يؤدي بالنتيجة إلى إبطاء سرعة المعالج [3]، وزيادة كمية السيليسيوم اللازم لصنع الدارة المتكاملة، وزيادة معدل استهلاكه للطاقة. وهكذا أخذت فكرة جديدة بالتبلور: قَصْر مجموعة تعليمات المعالج على التعليمات التي هناك موجب حقيقي لوجودها فقط. وظهر إلى الوجود مفهوم «ثوري» جديد في بنية الحواسيب: الحواسيب ذات مجموعة التعليمات الموجزة RISC (Reduced Instruction-Set Computer).

سنعرض فيما يلي لأهم مدرستين في تصميم الحواسيب ذات مجموعة التعليمات الموجزة، وهما مدرسة بيركلي Berkeley ومدرسة

3 بسبب زيادة عدد أدوار الساعة اللازم وسطياً لتنفيذ التعليمة الواحدة.

ستانفورد Stanford، ثم نلقي الضوء بعد ذلك على أهم الخصائص المشتركة في النوع من المعالجات، ونقارن مقارنة مجملة البنيان RISC بالبنيان التقليدي CISC.

2 مدرستا التصميم الرائدتان

2-1 مدرسة بيركلي

ظهر مصطلح RISC للمرة الأولى في التاريخ في جامعة بيركلي، وتحديداً في المقرر الذي يدرسه الأستاذ باترسون Patterson في بنية الحواسيب. وقد قام الأستاذ المذكور بطرح مشروع حول تصميم معالج ذي مجموعة تعليمات موجزة سُمي ببساطة RISC-I (انظر الشكل 1)، وبدأ هذا المشروع رسمياً في 1981/1/6، ورأى النموذج الأولي النور في 10/23 من العام نفسه!

وضع منفذو المشروع نصب أعينهم تصميم معالج أمثل ذي أداء عال، واعتمدوا منذ البداية الفرضيات الأربع التالية:

1 تنفيذ تعليمة في كل دور ساعة.

2 لجميع التعليمات الحجم نفسه.

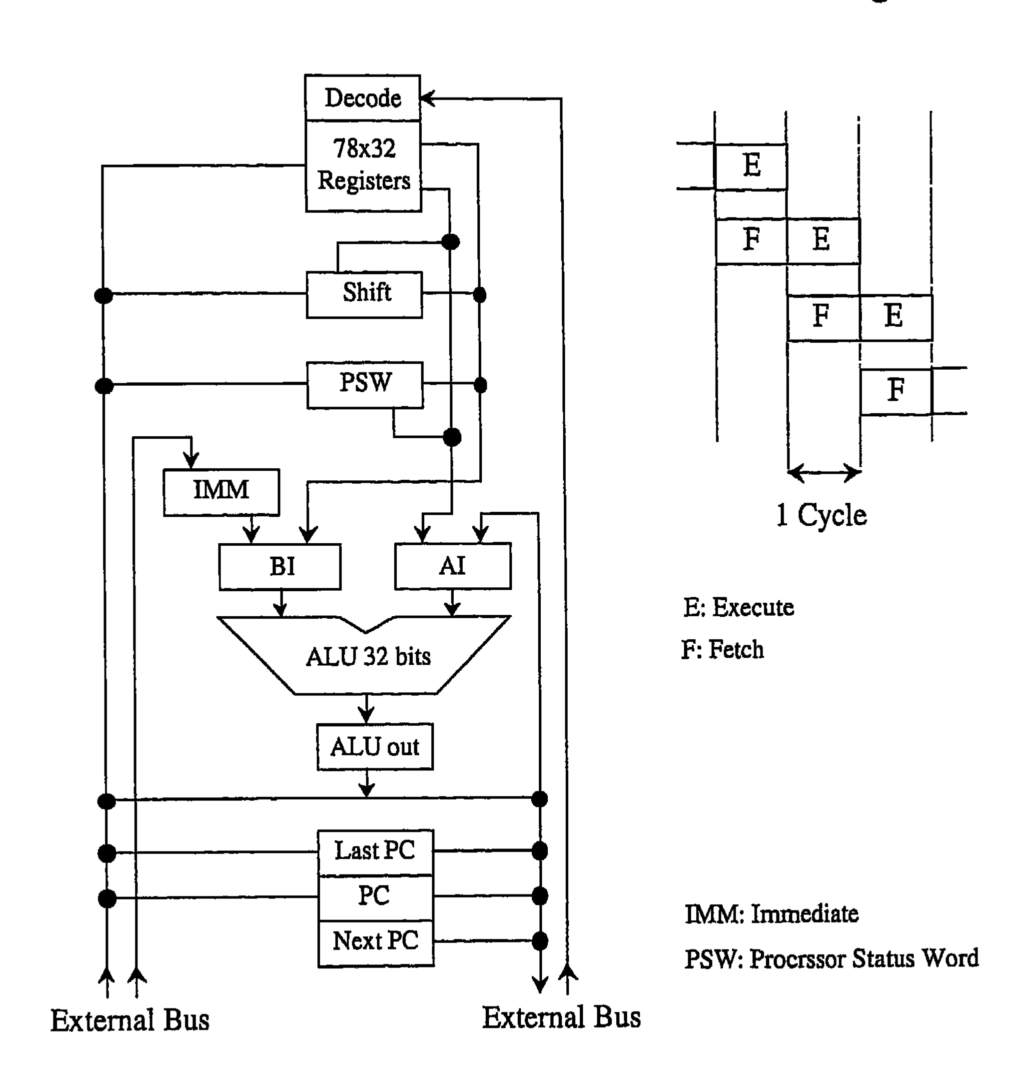
3 ضرورة تناسب لغة الآلة مع اللغات العالية المستوى.

4 يجري النفاذ إلى الذاكرة المركزية بواسطة تعليمتين فقط: Load من أجل شحن سجل داخلي بقيمة موجودة في موضع من الذاكرة؛ و Store من أجل تخزين قيمة سجل داخلي في موضع في الذاكرة. أما باقي التعليمات، فلا تتعامل إلا مع السجلات.

وقد استفاد المصممون من دراسة إحصائية أجريت على عدد كبير من البرامج المكتوبة بلغة C، وأعطت النتائج التالية:

– 80% من المتغيرات المحلية local المستخدمة في البرامج هي من النوع السُلّمي.

- 90% من بنى المعطيات المعقدة المستخدمة في البرامج تُمَثَّل بمتغيرات شمولية global.
- أغلب الإجرائيات لا تحتاج إلى أكثر من 6 محددات Arguments.
- لايتجاوز عدد طلبات الإجرائيات «المتداخلة»، الستة (6)، في 99% من الحالات.



الشكل 1: بنية المعالج RISC-I وقناة الموارد فيه.

نتيجة لذلك، ركّز المصممون اهتمامهم في استمثال التعامل مع المتغيرات المحلية التي يجري تخزينها في السجلات، وتحسين أداء عمليات طلب البرامج الجزئية والعودة منها إلى البرنامج الرئيسي بقدر الإمكان.

برهنت الاختبارات التي أجريت على النماذج الأولى من المعالج RISC-I بوضوح على صلاحية الفكرة. فعلى سبيل المثال، تبين بمقارنة RISC-I مع 68000 أن زمن تنفيذ برنامج معين هو وسطياً أسرع بـ 3.5±1.8 مرة في حالة الـ RISC-I، بالرغم من أن حجم هذا البرنامج، بعد ترجمته إلى لغة الآلة، هو وسطياً أصغر، بنسبة 0.9±0.2، في حالة الـ 68000. وقد شجعت هذه النتائج فريق المصممين في جامعة بيركلي على متابعة العمل على المشروع وإخراج المعالج RISC-II الذي يعتبر السلف الحقيقي لأحد أشهر معالجات RISC التجارية، وهو المعالج SPARC المستخدم في محطات العمل التي تسوّقها شركة SUN.

2-2 مدرسة ستانفورد

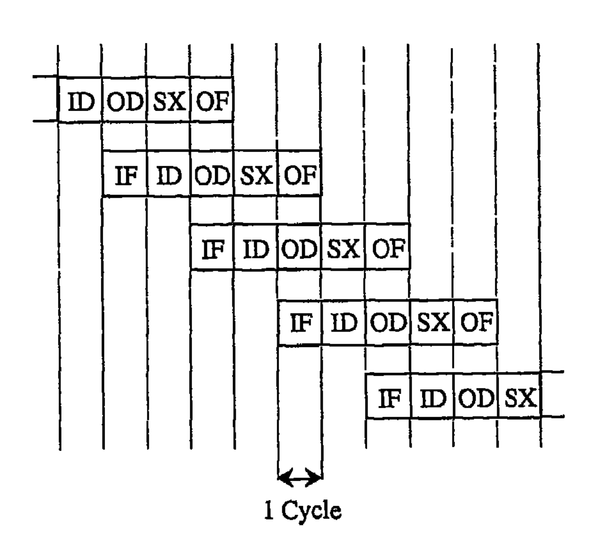
رائد بنيان المعالجات ذات مجموعة التعليمات الموجزة في جامعة ستانفورد هو الأستاذ هنسي Hennessy، الذي كان وراء تصميم معالج سُمي بـ MIPS (Machine without Interlocked Pipeline Stages)، وكان هنسي أيضاً وراء إنشاء الشركة التي حملت الاسم نفسه (MIPS). على غرار RISC-I، يقوم المعالج MIPS على أساس تبسيط مجموعة التعليمات (31 تعليمة مدونّة على 32 خانة). وإذا كان RISC-I يعتمد على المعالجة التواردية، فإن MIPS يدفع هذه التقنية إلى أوجها، وتقوم قناة المواردة فيه على خمس حلقات «متراكبة» تسمح بتنفيذ ثلاث تعليمات على التوازي (انظر الشكل 2).

أما التجديدة الأساسية في بنيان MIPS، فهي ضرورة دراسة

بنيانه دراسةً مشتركة مع مترجم لغة البرمجة العالية المستوى المعتمدة. ويجري تنفيذ أي برنامج على مرحلتين:

- المرحلة الأولى: الترجمة ثم التجميع.
- المرحلة الثانية: إعادة التنظيم. ويجري ذلك بواسطة برمجية خاصة تقوم بتحليل المُدَوَّنة الناتجة من المرحلة الأولى وإعادة تنظيمها بهدف استمثالها (مثلاً: حذف تعليمات NOP، حل مشاكل التضارب في استخدام الموارد المشتركة وخاصة قناة الموارِدة، ...).

إن هذه التقنية تسمح بتبسيط بنية المعالج وزيادة سرعة تنفيذ البرامج، ولكن على حساب زيادة تعقيد عملية الترجمة من اللغة العالية المستوى إلى لغة الآلة.



IF: Instruction Fetch

ID: Instruction Decode

OD: Operand Decode

SX: Store / eXecute

OF: Operand Fetch

الشكل 2: قناة الموارِدة في المعالج MIPS.

بعد النتائج المشجعة التي حصل عليها فريق العمل على هذا المعالج، قامت شركة MIPS بإنتاج المعالج التجاري MIPS-X، وبعد ذلك المعالجات R2000 و R3000.

3 الخصائص الأساسية لبنيان RISC

بعد أن استعرضنا في الفقرتين السابقتين أعمال المدرستين الأوليين في بنيان RISC، نأتي الآن إلى صياغة المبادئ والخصائص الأساسية المشتركة التي تميز هذا النوع من المعالجات.

3-1 مبادئ التصميم

رأينا آنفاً أن تصميم معالجات RISC ينطلق أساساً من التطبيق ليستنتج منه البنيان الملائم، ويجري ذلك عادة وفق الخطوات التالية:

1 إيجاد العمليات المنفذة بتواتر عال انطلاقاً من تحليل عدد كبير من البرمجيات المكتوبة في مجال التطبيق المعتمد.

2 استنتاج التعليمات اللازمة والكافية لتنفيذ العمليات الآنفة الذكر.

3 وضع بنيان معالج أمثل، منتظم قدر الإمكان، لتنفيذ التعليمات الآنفة الذكر والاستفادة إلى أقصى حد من تقنيات الموارِدة.

أما مبادئ التصميم الأساسية فيمكن إجمالها بما يلي:

1 تقليل عدد التعليمات ما أمكن، والاقتصار على التعليمات الضرورية في مجال التطبيق المعتمد. أما الوظائف المعقدة، فيُترك أمرها للمترجم.

2 جميع التعليمات تنفذ في حلقة وحيدة (قدر الإمكان).

3 يســتخدم عدد محدود من التعليمات للنـفاذ إلى الذاكرة - وفي أغلب الأحيان تكفي تعليمتان فقط: الشحن Load والتخزين Store. أما باقي التعليمات فلا تتعامل إلا مع السجلات.

4 زيادة عدد السجلات الداخلية.

5 تبسيط أنماط العنونة.

6 جميع التعليمات لها صيغة موحدة وثابتة الطول - الأمر الذي يسهل اعتماد تقنية الموازدة.

2-3 مجموعة التعليمات

ذكرنا آنفاً أن منهجية تصميم معالج من نمط RISC تعتمد على تحليل التطبيق المستهدف بغرض إيجاد مجموعة التعليمات المثلى. وبوجه عام، تحوي مجموعة التعليمات الأصناف التالية:

1 التعليمات الحسابية والمنطقية.

2 تعليمات التحكم في تسلسل تنفيذ البرنامج.

3 تعلميات النفاذ إلى الذاكرة.

إلى جانب هذه الأصناف تضاف عادة أصناف خاصة بالتطبيق المستهدف، شرط ألا تؤدي إلى تغيرات جوهرية في بنية المعالج وانتظامها.

يظهر الشكل 3 مجموعة تعليمات «صُوْرِيّة» formal توصي وزارة الدفاع الإمريكية DoD باعتمادها (أو بـ «الاستئناس» بها) عند تصميم معالج RISC جديد.

ABS	القيمة المطلقة		
ADD	جمع أعداد مع إشارة	ADDU	جمع أ اد بدون إشارة
DIV	قسمة أعداد مع إشارة	DIVU	قسمة أعداد بدون إشارة
MUL	ضرب أعداد مع إشارة	MULU	ضرب أعداد بدون إشارة
NEG	نفي عدد مع إشارة		
REM	باقي القسمة مع إشارة	REMU	باقي القسمة بدون إشارة
SUB	طرح أعداد مع إشارة	SUBU	طرح أعداد بدون إشارة
AND	عملية AND المنطقية	OR	عملية OR المنطقية
NOT	عملية NOT المنطقية	XOR	عملية XOR المنطقية
SLL	انزياح منطقي نحو اليسار	SDL	نفس SLL على كلمة مضاعفة
SRL	انزياح منطقي نحو اليمين	SDR	نفس SRL على كلمة مضاعفة
SRA	انزياح حسابي نحو اليمين	SDA	نفس SRA على كلمة مضاعفة
ROL	دوران نحو اليسار	RDL	نفس ROL على كلمة مضاعفة
ROR	دوران نحو اليمين	RDR	نفس ROR على كلمة مضاعفة
BRA	قفز غير مشروط	BRC	قفز مشروط
JMP	قفز مع رابط	TRAP	مقاطعة برمجية
CALL	استدعاء برنامج فرعي	RET	العودة من برنامج فرعي
LDBS	شحن ثمانيّة مع إشارة	LDBU	شحن ثمانيّة بدون إشارة
LDHS	شحن رباعيّة مع إشارة	LDHU	شحن رباعيّة بدون إشارة
LDW	شحن كلمة		
STB	تخزين ثمانيّة		
STH	تخزين رباعيّة		
STW	تخزين كلمة		

الشكل 3: مجموعة تعليمات RISC صورية.

1-2-3 صيغة التعليمات

من الخصائص الأساسية التي تميز معالجات RISC اعتماد مصوغة format موحَّدة وثابتة الحجم لكافة التعليمات، يكون حجمها عادة

مساوياً لعرض الكلمة في الذاكرة (32 خانة في أغلب الأحيان). تقسم المصوغة إلى عدد من الحقول التي تحدد عدد المعاملات Operands ومحتوياتها وأنماط العنونة (انظر مثلاً مصوغة التعليمات في المعالج RISC-I في الشكل 4). ويؤدي الانتظام في صياغة التعليمات إلى سهولة تنفيذها عبر قناة الموارِدة.

	SCC			IM		
OPCODE		DEST	SRC1		SRC2	Format 1
7	1	5	5	1	13	

	SCC			
OPCODE		DEST	IMM	Format 2
7	1	5	19	

الشكل 4: مصوغة التعليمات في المعالج RISC-I.

3-2-2 استخدام السجلات

تستخدم تعليمات المعالجات RISC السجلات بكثرة، والسبب في ذلك بيِّن: فعنق الزجاجة في أي معالج يكمن في النفاذ إلى الذاكرة (زمن استجابة الذاكرة يساوي وسطياً 50-200ns في حين أن زمن تنفيذ تعليمة تحوي محددات مدوّنة في سجلات يساوي وسطياً 20-50ns). وهكذا، فإن زيادة سرعة العمل تتعلق بالحد من عدد مرات النفاذ إلى الذاكرة، ويكون ذلك بالإكثار من استخدام السجلات الداخلية لتخزين المعطيات أثناء تنفيذ البرنامج، وهذا ما يدعو بداهة إلى زيادة عدد تلك السجلات.

3-3 انتظام البنية الداخلية

درسنا في فصل سابق كيف يمكن أن نجزئ أي معالج إلى جزأين وظيفيين: الجزء التنفيذي العامل (الذي يطلق عليه عادة اسم ممر

المعطيات)، والجزء المتحكم المسؤول عن سَلْسَلَة التعليمات، وعن جلب التعليمات وفك ترميزها.

3-3-1 الجزء التنفيذي

يمكن تمثيل الجزء التنفيذي، في معالج ذي سجلات، كما في الشكل 5، الذي يظهر أيضاً الصياغة التقليدية لقناة الموارِدة المطبقة على هذه البنية بمراحلها الأربع: جلب التعليمة وفك ترميزها؛ وقراءة متغيرات الدخل؛ وتنفيذ العملية؛ وتخزين متغيرات الخرج.

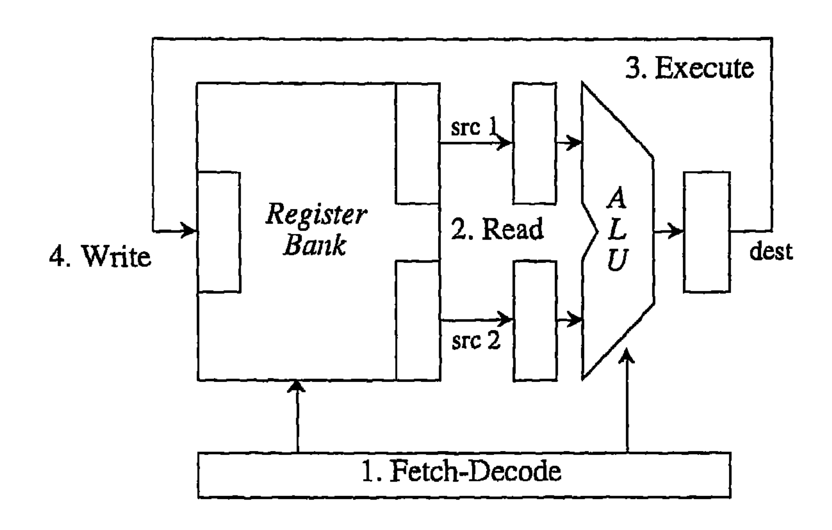
تعتمد معظم معالجات RISC بوجه عام هذه البنية التي تسمح بتنفيذ العمليات تنفيذاً منتظماً عبر قناة الموارِدة. أما حالات «التضارب» (مثال: تعليمة تستخدم في الدخل سجلاً يجري تغيير قيمته في خرج التعليمة التي تسبقها مباشرة) فتحل، كما ذكرنا آنفاً، إما بتعديل البنية الصلبة (في المعالج RISC-I والمعالجات المبنية عليه) أو بإعادة تنظيم المدوّنة الناتجة (في المعالج MIPS والمعالجات المبنية عليه).

3-3-2 فك ترميز التعليمة

تؤدي بساطة مصوغة التعليمات في معالجات RISC إلى سهولة فك ترميزها، فتقتصر هذه العملية في معظم الحالات على تخزين المعاملات في السجلات المخصصة لها، ثم إصدار الأوامر المناسبة إلى الجزء التنفيذي بناء على قيمة رمز التعليمة.

3-3-3 بنيان هارڤرد

يؤدي تنفيذ التعليمات، عبر قناة الموارِدة، عملياً إلى إنجاز تنفيذ تعليمة كاملة في كل حلقة ساعة. نقول عملياً لأن التضارب على القناة قد يؤدي إلى بعض الاضطراب في انتظام التنفيذ.



Read	Execute	Store		
Fetch	Read	Execute	Store	
	Fetch	Read	Execute	Store
		Fetch	Read	Execute

1 Cycle

الشكل 5: الجزء التنفيذي في معالج ذي سجلات.

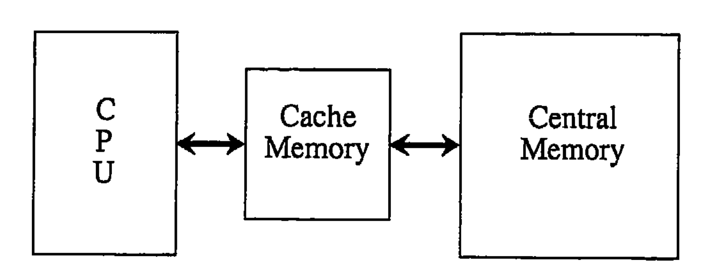
بيد أن هناك حالة أخرى تؤدي إلى اضطراب عمل القناة: النفاذ إلى الذاكرة. فزمن النفاذ إلى الذاكرة، كما ذكرنا آنفاً، أطول من زمن تنفيذ التعليمة في وحدة الحساب والمنطق، الأمر الذي يعني ضرورة «إيقاف» الموارد مؤقتاً من أجل تنفيذ عمليات النفاذ إلى الذاكرة.

يحتاج تنفيذ تعليمتي الشحن والتخزين غالباً إلى حلقتي ساعة؛ ولما كانت الإحصاءات تدل على أن هذه التعليمات تُمثِّل وسطياً 20–25% من مجمل التعليمات المنفذه في البرنامج، فإنها تؤدي إلى تأخير ملحوظ في زمن التنفيذ الإجمالي.

أحد الحلول المقترحة لهذه المشكلة هو اعتماد بنيان هارفرد (انظر الفصل الخامس، الفقرة 5-2)، أي بناء معالج ذي مسريين مستقلين: أحدهما للتعليمات، ويسمح بالنفاذ إلى ذاكرة البرنامج؛ والثاني للمعطيات، ويسمح بالنفاذ إلى ذاكرة المعطيات. يسمح هذا البنيان بإيجاد حل للمشكلة الآنفة الذكر وتحسين أداء المعالج، ولكن على حساب زيادة حجم الكيان الصلب.

3-3-4 الذواكر الخابية

تسمح تقنية الذاكرة الخابية Cache Memory بزيادة سرعة النفاذ الوسطية إلى الذاكرة المركزية، عن طريق استخدام ذاكرة سريعة جداً ولكن صغيرة الحجم (حتى تبقى تكلفتها معقولة) توضع بين المعالج والذاكرة المركزية (انظر الشكل 6). تكون هذه الذاكرة عادة أسرع بـ 5 إلى 20 مرة من الذاكرة الاعتيادية؛ وأصغر بـ 50 إلى 1000 مرة من الذاكرة المركزية.



الشكل 6: الذاكرة الخابية.

نضع في هذه الذاكرة البينية، المسماة عادة بالخابية، التعليمات و/أو المعطيات التي يستخدمها المعالج بكثرة، وفقاً لمفهوم الموضعية الزمانية والمكانية:

- الموضعية الزمانية: ثمة احتمال كبير في نفاذ المعالج مرات متتالية إلى المواضع نفسها في الذاكرة.

- الموضعية المكانية: إذا نفذ المعالج إلى موضع ما في الذاكرة، فثمة احتمال كبير في نفاذه إلى مواقع قريبة في المرات القادمة.

وهكذا يكفي أن نخزن في الذاكرة الخابية آخر مجموعة «متراصة» من التعليمات (خابية التعليمات) و/أو المعطيات (خابية المعطيات) التي نفذ المعالج إليها. ومن الواضح أن استخدام تقنية الذاكرة الخابية في معالج له بنيان هارفرد يعطي أداء أفضل.

نسمي معدل وجود تعليمة/معطاة يطلبها المعالج في الذاكرة الخابية بنسبة النجاح - وقد تصل هذه النسبة إلى 90% في حالة بعض الاستراتيجيات الناجعة التي تتحكم في تبديل محتوى الخابية وفق الوضعية الزمانية والمكانية اللحظية.

مثال:

زمن النفاذ إلى الذاكرة المركزية 200ns؛

نسبة النجاح 90%؛

زمن النفاذ إلى الذاكرة الخابية 40ns؛

زمن الاستجابة «الظاهري»:

200*0.1 + 40*0.9 = 56ns

أي إن سرعة النفاذ الوسطية قد زادت زهاء 3.5 مرة.

نذكر أخيراً أن صغر حجم الذاكرة الخابية وعلاقتها الوثيقة بوحدة المعالجة المركزية يدفع المصممين إلى إدماجها مع المعالج في الدارة المتكاملة نفسها (مثلاً في MIPS-X).

4 البنيان RISC في مقابل CISC

لنعرض الآن بإيجاز بعض النتائج التي يؤدي إليها اعتماد بنيان RISC بالمقارنة بالبنيان التقليدي المعالجات CISC.

4–1 زيادة حجم البرامج بلغة الآلة

تدعو فلسفة RISC، كما بيّنا آنفاً، إلى قَصْر مجموعة تعليمات المعالج على ما هو ضروري فقط. وقد رأينا أن ذلك يؤدي إلى زيادة فعالية المعالج وسرعة أدائه في التطبيقات المستهدفة. ولكن، من جهة أخرى، يؤدي ذلك إلى كبر حجم البرنامج المترجم من لغة برمجة عالية المستوى إلى لغة الآلة في المعالجات من نمط RISC بالمقارنة بمثيلاتها من نمط CISC، وذلك بسبب حاجة التعليمات عالية المستوى إلى عدد أكبر من تعليمات الآلة عند ترجمتها.

وعموماً، تُقدر هذه الزيادة في الحجم بنحو 20%، وهي تبقى مقبولة إذا ما أخذنا بعين الاعتبار تحسين الأداء بنحو 3–6 مرات.

4–2 النفاذ إلى الذاكرة

يؤدي النفاذ إلى الذاكرة المحدود بتعلمتي الشحن والتخزين إلى تبسيط مجموعة التعليمات وزيادة انتظام العمليات المتواردة. ولكن ذلك يكون عادة على حساب فعالية بعض الوظائف، مثل بعض العمليات البيانية، ومثل البحث عن سلاسل من المحارف في نص واستبدالها، وماشابه...

4–3 زيادة تعقيد المترجمات

إن تبسيط مجموعة التعليمات وانتظام التنفيذ يؤديان إلى تعقيد أكبر في خوارزميات الترجمة. وفي بعض الأحيان، تضع

معالجات RISC مطوري المترجمات أمام تحد كبير إذا هم أرادوا تطوير مترجم كفء وفعال. ويزيد من جدية المشكلة التناقص المطرد في البرمجيات/الإجرائيات المكتوبة بلغة المجمع بالمقارنة بتلك المكتوبة بلغة عالية المستوى، وهذا ما يزيد الحاجة إلى وجود مترجم كفء.

لهذا السبب، يعتمد المصممون اليوم منهجية تقوم على تطوير المترجمات «على التوازي» مع تصميم المعالج؛ بحيث يجري تعديل أحدهما بما يوافق حاجات الآخر؛ مما يعطي في النهاية معالجاً ذا بنيان أمثلي يملك مترجماً قديراً وفعالاً.

5 خاتمة

برغم المصاعب المذكورة آنفاً، فإن معالجات RISC هي اليوم واعدة أكثر من أي وقت مضى. وهي تلائم المعالجات المصممة لتطبيقات متخصصة أكثر من المعالجات العامة الاستخدام، وإن كانت تستخدم بكثرة في محطات العمل العصرية (مثال: المعالج SPARC في محطات العمل SUN).

وفي جميع الأحوال، ولو أن الجدال بين أنصار الـ RISC وأنصار الـ CISC لم يحسم بعد لمصلحة أي من الطرفين، فإن المبادئ التي وضعها أصحاب مدرسة RISC قد أخذت تؤثر، ولو جزئياً، في تصميم المعالجات العصرية، فبعض معالجات الإشارة الرقمية، مثل TMS320C40 من شركة Texas Instruments، برغم سعة مجموعة تعليماته، يتبنى بعض مبادئ RISC الأساسية، مثل المعالجة التواردية، وانتظام مصوغة التعليمات، الخ... وكذلك حال معالجات ينتمي أسلافها بصراحة إلى نمط CISC، مثل Pentium من شركة Intel الذي تأثر في تصميمه بالمبادئ الآنفة الذكر لدرجة أن بعضهم يصنّفه كمعالج RISC!

الملاحق

الملحق الأول

تذكرة بأنظمة العد والترميز

1 مقدمة

يمكن تمثيل أي عدد بسلسلة من الأرقام 0,1,2,...,b-1، ونسمي العدد b أساس العد. ففي التمثيل العشري يكون b=10 وفي العد الثماني يكون b=8، أما في العد الاثناني Binary فيكون b=2.

وكما في التمثيل العشري المعروف، لكل رقم قيمة تابعة لترتيبه في السلسلة وتابعة لترتيبه في العدد. فالرقم 5 في العدد 2151 له القيمة 50 في التمثيل العشري والقيمة 40 في التمثيل الثماني. وتعطى قيمة كل رقم x بـ $x.b^{i-1}$، حيث b هو الأساس و i ترتيب الرقم في العدد محسوباً من اليمن إلى اليسار. وتُمثّل الأعداد الكسرية باستخدام القوى السالبة للأساس. فالعدد 146.17 له في التمثيل الثماني مثلاً القيمة:

$1*8^2+4*8+6*8^0+1*8^{-1}+7*8^{-2}$

نلاحظ أنه لا يمكن لرقم ما في تمثيل أساسه b أن يكون أكبر من b-1، وعليه تمثل الأعداد في التمثيل الاثناني باستخدام رقمين هما الصفر والواحد.

2 الخواص العامة لتمثيل الأعداد

ليكن العدد $A = a_n a_{n-1} ..... a_1 a_0$ في نظام العد ذي الأساس b، يكون لدينا عندئذ $a_i \in \{0,1,....,b-1\}$، وتكون قيمة هذا العدد مساوية لـ $A = \sum_{i=0}^{n} a_i b^i$.

يمكن أن نكوّن من مجموعة من n خانة في نظام عد أساسه b b^n عدداً مختلفاً، قيمها محصورة بين الصفر والـ $b^n - 1$.

لنلاحظ أن a_i هي بواقي القسمة المتتالية للعدد A في الأساس b، أي:

$$A = b(a_n.b^{n-1} + a_{n-1}.b^{n-2} + a_{n-2}.b^{n-3} + ... + a_1.b^0) + a_0 = b.Q_1 + a_0$$

حيث:

$$Q_1 = b(a_n.b^{n-2} + a_{n-1}.b^{n-3} + ... + a_2.b^0) + a_1 = b.Q_2 + a_1$$
$$Q_2 = b(a_n.b^{n-3} + a_{n-1}.b^{n-4} + ... + a_3.b^0) + a_2 = b.Q_3 + a_2$$

...

...

$$Q_{n-1} = b.a_n + a_{n-1} = b.Q_n + a_{n-1}$$
$$Q_n = b.0 + a_n = b.0 + a_n$$

نلاحظ أن عملية القسمة هذه يمكن إجراؤها في أي نظام عد، ومن ثمَّ تتيح عملية القسمة الانتقال من أي تمثيل بأساس b إلى تمثيل آخر بأساس h. لنجد مثلاً تمثيل العدد العشري 198 في الأساس 7:

198 = (7*28) + 2
28 = (7*4) + 0
4 = (7*0) + 4

ومن ثم يكون: (198)10 = (402)7.

سنهتم في هذا الفصل بنظام العد الاثناني نظراً لما يتمتع به من خواص عملية هامة تتمثل في المقام الأول في إمكان تمثيله كهربائياً؛ وفي قابلية تنفيذ ذلك التمثيل تقانياً بوثوقية عالية وكلفة

منخفضة؛ وكذلك في إمكان التمثيل المباشر للعلاقات أو الظواهر التي تأخذ أحدى حالتين، كالصواب والخطأ؛ والأسود والأبيض؛ ووصول عربة إلى المحطة أو لا؛ الخ...

3 التمثيل الاثناني

تُكتب الأعداد الاثنانية على شكل سلسلة من الأصفار والوُحدان، كما في العدد 1101 الذي يمثل العدد العشري 13:

$1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 13$

1-3 الانتقال من التمثيل الاثناني إلى التمثيل العشري

نحصل على التمثيل العشري لعدد اثناني $N_2 = a_n\, a_{n-1} \ldots\ldots a_1\, a_0$ حيث $a_i \in \{0,1\}$، باتباع العلاقة:

$N_{10} = a_n.2^n + a_{n-1}.2^{n-1} + \ldots\ldots + a_1.2 + a_0.2^0$

مثال:

ليكن العدد الاثناني $N_2 = 111001$. باستخدام العلاقة السابقة نجد:

$$N_{10} = 1*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$$
$$= 32 + 16 + 8 + 1 = 57$$

2-3 الانتقال من التمثيل العشري إلى التمثيل الاثناني

كما ذكرنا آنفاً، نجد التمثيل الاثناني للعدد العشري باستخدام عملية التقسيم المتتالي للبواقي.

مثال:

$N_{10} = 57$

نجري عملية القسمة المتتالية، فنجد:

57	2					
1	28	2				
	0	14	2			
		0	7	2		
			1	3	2	
				1	1	2
					1	0

ومن ثم: $N_2 = 111001$.

يمكن إيجاد الترميز الاثناني لعدد عشري بتجزئته إلى رزم، على النحو التالي:

- نكتب العدد العشري بالشكل $A = Q.2^p+R$، حيث $R < 2^p$.
- نجدالتمثيل الاثناني للعدد R الذي يمثل المراتب الدنيا.
- نتابع البحث عن تمثيل الجزء $Q.2^p$ بتجزئته بالطريقة نفسها.

<u>مثال:</u>

المطلوب إيجاد التمثيل الاثناني للعدد العشري 263.

نلاحظ أن:

$$263 = 32 * 2^3 + 7$$
$$= (4 * 2^3) * 2^3 + 7 = 4*8^2 + 0*8^1 + 7*8^0$$

ولما كان التمثيل الاثناني للعدد 7 هو 111، وللعدد 0 هو 000، وللعدد 4 هو 100، كان تمثيل العدد 263 هو 100000111.

3-3 العمليات الحسابية في النظام الاثناني

أ- الجمع: انطلاقاً من العلاقات التالية:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$

نقوم بعملية الجمع كما في جمع الأعداد في النظام العشري.

<u>مثال:</u>

```
  6 :  0110
+10 :  1010
-----------
 16 : 10000
```

ب- الطرح: انطلاقاً من العلاقات التالية:

0 - 0 = 1
0 - 1 = 1*
1 - 1 = 0
1 - 0 = 1

(حيث تشير * إلى وجود استعارة)
نقوم بعملية الطرح كما في طرح الأعداد في النظام العشري.

مثال:

```
 9 : 1001
-6 : 0110
---------
 3 : 0011
```

أما إذا كان حاصل الطرح سالباً، فنعود بالمسألة إلى تمثيل الأعداد الاثنانية السالبة في الحاسوب.

مثال:

```
 6 : 0110
-9 : 1001
---------
    11101
```

حيث تشير الخانة الاثنانية اليسرى، التي تساوي الواحد، إلى أن العدد الناتج سالب.

يمكن معرفة القيمة المطلقة لحاصل الطرح (كما في حالة التمثيل

العشري) بطرح الحاصل من العدد 10000. وتكافىء عملية إيجاد معكوس القيمة تبديل كل 1 بـ 0 وكل 0 بـ 1 في حاصل الطرح، ثم إضافة 1 إلى الحاصل، أي: 00010 + 1 = 00011. تسمى هذه العملية بعملية المتمم الاثناني Two's Complement.

ج- الضرب: انطلاقاً من العلاقات التالية (جدول الضرب الاثناني):

0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1

نقوم بعملية الضرب كما في ضرب الأعداد العشرية.

<u>مثال</u>:

```
 7 :   111
 2 :   010
-----------
       000
      111
     000
-----------
14 : 01110
```

د- القسمة:

نقوم بعملية القسمة كما في قسمة الأعداد العشرية.

<u>مثال</u>: 12/3 = 4

```
1100 | 011
11     100
--
  00
  00
  --
   0
```

4 مفاهيم أساسية في الترميز العددي

يقصد بالترميز العملية التي تستخدم تمثيلاً عددياً معيناً، إذ يمكن لعدد ما أن يعبر عن قيمة أو مقدار فيزيائي أو حالة جملة فيزيائية.

يوجد عموماً نوعان من الترميز: الترميز الموزون والترميز غير الموزون.

1-4 الترميز الموزون للأعداد العشرية

في هذا النوع من الترميز يلحق بكل خانة وزن محدد، مثل إعطاء الوزن 1 للخانة الأولى و 2 للخانة الثانية و 4 للخانة الثالثة وأخيراً 2 للخانة الرابعة، وذلك في حالة تمثيل الأعداد العشرية على أربع خانات اثنانية، ومن ثم يكون الترميز 1101 مكافئاً للعدد العشري:

1*2 +1*4 +0*2 +1*1 = 7

1-1-4 الترميز الاثناني للأعداد العشرية

وهو الترميز المشهور بـ(Binary-Coded Decimal) BCD المستخدم لترميز الأرقام العشرية من 0 إلى 9.

هنا يكون وزن الخانة الأولى $2^0 = 1$، والثانية $2^1 = 2$ ، والثالثة $2^2 = 4$، وأخيراً الرابعة $2^3 = 8$. ومنه نحصل على الجدول:

0000 = 0	0101 = 5
0001 = 1	0110 = 6
0010 = 2	0111 = 7
0011 = 3	1000 = 8
0100 = 4	1001 = 9

فلتمثيل العدد العشري 257 مثلاً نجد:

257 : 0010 0101 0111

يُسمي هذا الترميز أيضاً بالترميز الموزون 8.4.2.1.

2-1-4 التراميز الموزونه الذاتية التتميم للأعداد العشرية

يقصد بذاتية التتميم أن للرقم a ترميزاً هو متمم ترميز الرقم 9-a.

لنأخذ مثالاً على ذلك ترميز هارفرد، حيث تشير الأرقام 1,2,4,8، إلى وزن الخانة في الترميز، مع الانتباه إلى أن الرقمين 1,2 اللذين وضع فوقهما - يجب أن يطرح ناتجهما عند حساب الرقم المقابل للترميز.

- تمثيل هارفرد Harvard

$8\ 4\ \bar{2}\ \bar{1}$		$8\ 4\ \bar{2}\ \bar{1}$	
1 0 1 1	= 5	0 0 0 0	= 0
1 0 1 0	= 6	0 1 1 1	= 1
1 0 0 1	= 7	0 1 1 0	= 2
1 0 0 0	= 8	0 1 0 1	= 3
1 1 1 1	= 9	0 1 0 0	= 4

ويمكن إيجاد قيمة أي عدد في هذا التمثيل كما في المثال التالي:

1001 = 1*8 + 0*4 - 0*2 - 1*1 = 7

2-4 الترميز غير الموزون

في هذا النوع من الترميز، لا أهمية للخانة في حد ذاتها وإنما الأهمية لجمل الترميز. لنأخذ مثالاً على ذلك الترميز المزاح.

- الترميز المزاح بمقدار 3 Excess-3

هو ترميز الأعداد العشرية بعد إضافة 3 إلى كل منها. لهذا الترميز خاصية كونه ذاتي التميم، ويحتوي دائماً على خانة تساوي الواحد.

1000	= 5	0011	= 0
1001	= 6	0100	= 1
1010	= 7	0101	= 2
1011	= 8	0110	= 3
1100	= 9	0111	= 4

3-4 الترامیز العددیة غیر العشریة

وهي ترامیز ثابتة تمثل الأعداد تمثیلاً مستقلاً عن ترمیز الأرقام العشریة، بمعنى آخر، یحدد كل ترمیز القواعد الخاصة به التي تسمح بإیجاد ترمیز أي عدد.

- الترمیز الطبیعي Natural Binary Code

وهو الترمیز الطبیعي الموزون المعروف، حیث للخانة n الوزن 2^{n-1}. مثال ذلك تمثیل العدد العشري 126:

$$126 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 = 111\ 1110$$

- الترمیز المنعكس أو ترمیز Gray

وهو ترمیز یختلف فیه كل مستویین متتالین بخانة اثنانیة واحدة، أي إن كل مستویین یكوّنان تجاوراً، وهذا ما یسهل عملیة الكشف عن الأخطاء، وخاصة في حالة محسّات الموضع العددیة (التي تعتمد مثل هذا الترمیز)، إذ ننتقل دائماً من وضع إلى وضع مجاور، أي من ترمیز إلى ترمیز مجاور.

نلاحظ هنا التناظر (في حالة أربع خانات كما هو ظاهر) بالنسبة للمحور الذي یفصل بین المستویین 7 و 8؛ وكذلك بالنسبة للمحاور التي تفصل بین المستویین 1 و 2؛ 5 و 6...

تسمح خاصة التناظر هذه بإیجاد الأعداد العلیا. فمن أجل ترمیز معكوس ذي n خانة إثنانیة، یمكن أن نلاحظ تساوي الخانات n-1 الدنیا حول محور التناظر الذي یفصل بین المستویین $2^{n-1} - 1$ و 2^{n-1}؛ وكذلك تساوي الخانات n-2 الدنیا حول المحاور $2^{n-2} - 1$ و 2^{n-2}؛ وكذلك بین $2^{n-2} + 2^{n-1} - 1$ و $2^{n-2} + 2^{n-1}$

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

مثال: أوجد ترميز Gray للعدد 124

نلاحظ أن $2^7 = 128 > 124 > 64 = 2^6$ وعليه نحتاج إلى 7 خانات اثنانية لترميز هذا العدد. لنلاحظ أن محور التناظر يقع بين العدد $128 = 2^7$ و $127 = 1 - 2^7$ أي إن ترميز هذين العددين متساو في الخانات السبع الأولى. ولأن ترميز العدد 127 هو 1000000 يكون ترميز العدد 128 هو 11000000. وبملاحظة أن $3 = 124 - 127$ يكون ترميز العدد 124 مساوياً ترميز العدد $3 + 128 = 131$ في الخانات السبع الدنيا، وهو من ثم: $1000000 + 01 = 1000010$.

4-4 التحويل بين الترميز الطبيعي والترميز المنعكس

لنضع في جدول واحد الترميز الطبيعي الاثناني والترميز

المنعكس لمجموعة الأعداد التي يحتاج ترميزها مثلاً إلى خمس خانات اثنانية:

edcba	ηδγβα
00000	00000
00001	00001
00010	00011
00011	00010
00100	00110
00101	00111
00110	00101
00111	00100
01000	01100
01001	01101
01010	01111
01011	01110
01100	01010
01101	01011
01110	01001
01111	01000
10000	11000
10001	11001
...	...

1-4-4 التحويل من الترميز الطبيعي إلى الترميز المنعكس

نلاحظ من الجدول السابق وجود علاقات بين e d c b a (أعمدة الترميز الطبيعي) و η δ γ β α (أعمدة الترميز المنعكس) على النحو التالي:

$$\alpha = a \oplus b$$
$$\beta = b \oplus c$$
$$\gamma = c \oplus d$$
$$\delta = d \oplus e$$
$$\eta = e$$

حيث يقصد بالعملية ⊕ العلاقة: $\alpha \oplus \beta = \alpha.\overline{\beta} + \overline{\alpha}.\beta$.

يمكن تعميم هذه العلاقات في حالة الترميز على n خانة اثنانية. لتكن a_i أعمدة الترميز الطبيعي و α_i أعمدة الترميز المنعكس $(i=1,...n)$.

$$\alpha_1 = a_1 \oplus a_2$$

...

$$\alpha_i = a_i \oplus a_{i+1}$$

...

$$\alpha_n = a_n$$

مثال: إيجاد التمثيل المنعكس للعدد 124

الترميز الطبيعي: 124 = 1111100.

ومنه نجد:

$$\alpha_1 = 0$$
$$\alpha_2 = 1$$
$$\alpha_3 = 0$$

...

$$\alpha_6 = 0$$
$$\alpha_7 = 1$$

على هذا فالترميز المنعكس لـ 124 هو 1000010.

2-4-4 التحويل من الترميز المنعكس إلى الترميز الطبيعي

بملاحظة أنه إذا كان $x = y \oplus z$ فإن $y = x \oplus z$، نجد بالعودة إلى العلاقات السابقة:

$$a_1 = \alpha_1 \oplus \alpha_2 \oplus \alpha_3 \; .... \; \oplus \alpha_n$$
$$a_2 = \alpha_2 \oplus \alpha_3 \; .... \; \oplus \alpha_n$$
$$a_3 = \alpha_3 \oplus \alpha_4 \; .... \; \oplus \alpha_n$$

...

$$a_n = \alpha_n$$

5 ترامیز كشف الأخطاء

تنقل المعلومات بين حاسوب وآخر مثلاً عن طريق كبل يصل مباشرة بين الحاسوبين مباشرة أو عن طريق شبكة. والنقل المباشر بين الحاسوبين يكون إما «تسلسلياً»: ترسل الخانات الاثنانية من الحاسوب الأول ويستقبلها الحاسوب الثاني خانة بعد خانة؛ أو يكون «تفرعياً»: يرسل العدد دفعة واحدة.

في كلتا الحالتين، من المهم نقل هذه المعلومات نقلاً صحيحاً. ولكن ذلك يصعب بسبب الضجيج الكهربائي أو التشوه الذي قد يصيب الإشارة... لذا يجب توفير وسيلة تسمح بكشف الخطأ عند حدوثه وتصحيحه إن أمكن.

يمكن كشف الخطأ عند ترميز العدد (أو الإشارة) في عدد من الخانات (أو الأعمدة) أكبر من العدد اللازم، بحيث تعطى الأعمدة الزائدة دور المساعدة في كشف الأخطاء. مثلاً، يمكن إضافة عمود التكافؤ parity الذي نضع فيه القيمة 1 إذا كان عدد الخانات المساوية للواحد في الترميز زوجياً والقيمة 0 إذا كان ذلك العدد فردياً، مع تجنب الترميز 0000 الذي قد يستخدم للدلالة على عطل ما في النظام المرسل للمعلومات. فلو أخذنا على سبيل المثال الترميز المزاح بمقدار 3 وأرفقنا به عمود التكافؤ لكان لدينا:

عمود التكافؤ	الترميز المزاح بمقدار 3	
1	0011	0
0	0010	1
1	0101	2
1	0110	3
0	0111	4
0	1000	5
1	1001	6
1	1010	7
0	1011	8
1	1100	9

عند استقبال المعلومات، نقوم بإعادة حساب خانة التكافؤ ومقارنتها بالخانة المستَقبَلة للاستدلال على وجود خطأ أم لا. مثلاً أذا استقبلنا العدد 0100 وكانت خانة التكافؤ الملحقة به تساوي 0 كان هذا مؤشراً إلى صحة الاستقبال؛ أما إذا كانت خانة التكافؤ المستَقبَلة تساوي 1 فعندها يمكن توقع وجود خطأ في استقبال الخانات. غير أن ذلك لا يسمح بتصحيح الخطأ. من جهة أخرى، لا تسمح هذه الطريقة بكشف خطأ مضاعف، ولكن احتمال الخطأ المركب أقل بكثير من احتمال وجود خطأ في خانة واحدة.

- <u>ترميز هامنغ</u> Hamming

في هذا الترميز، ترمز الأعداد في عدد من الخانات أكبر بثلاثة أعمدة من العدد اللازم. تكوّن بعض هذه الاعمدة الترميز الاثناني العادي، وتكوّن الأعمدة فيما بينها مجموعات للتوثق من صحة الترميز وتصحيحه.

لنأخذ مثلاً ترميز الأعداد من 0 إلى 15:

	p_1	p_2	x_3	p_4	x_5	x_6	x_7
0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1
2	0	1	0	1	0	1	0
3	1	0	0	0	0	1	1
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1
6	1	1	0	0	1	1	0
7	0	0	0	1	1	1	1
8	1	1	1	0	0	0	0
9	0	0	1	1	0	0	1
10	1	0	1	1	0	1	0
11	0	1	1	0	0	1	1
12	0	1	1	1	1	0	0
13	1	0	1	0	1	0	1
14	0	0	1	0	1	1	0
15	1	1	1	1	1	1	1

حيث تكوّن مجموعة الأعمدة $x_3x_5x_6x_7$ الترميز الطبيعي، في حين

تكوّن الأعمدة $p_1x_3x_5x_7$ و $p_2x_3x_6x_7$ و $p_4x_5x_6x_7$ ، حيث $p_1=x_3 \oplus x_5 \oplus x_7$ و $p_2=x_3 \oplus x_6 \oplus x_7$ و $p_4=x_5 \oplus x_6 \oplus x_7$ مجموعات التحقق التي تضمن عدداً زوجياً من الوحدان. ومن ثم، ففي حالة حدوث خطأ أثناء النقل، لن يكون الشرط السابق محققاً وسيكون بالإمكان كشف مكان الخطأ.

لنأخذ مثالاً على ذلك ترميز العدد 9 وهو 0011001. لنفترض الآن وجود خطأ في الخانة الأخيرة بحيث يكون لدينا 0011000.

نجد من مجموعات التحقق ما يلي:

$p_1x_3x_5x_7$ $\quad 0\oplus 1\oplus 0\oplus 0=1$

$p_2x_3x_6x_7$ $\quad 0\oplus 1\oplus 0\oplus 0=1$ $\quad \rightarrow \quad 7:111$

$p_4x_5x_6x_7$ $\quad 1\oplus 0\oplus 0\oplus 0=1$

تدل هذه القيمة على مكان الخطأ، وهو الخانة الأخيرة التي يجب أن تكون قيمتها مساوية للواحد.

الملحق الثاني

التنفيذ التقاني للمؤثرات المنطقية

1 مقدمة

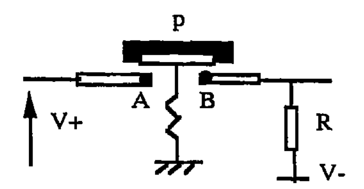
يتطلب الاستخدام العملي لمختلف المؤثرات المنطقية التي رأيناها في الفصل السابق أدوات ووسائل تقنية تسمح بإيجاد مكافئات مادية لها. بعض التقنيات بقي على حاله دون تطور كما هو الحال في تقنيات الأزرار والقواطع الكهربائية، على حين لم ينقطع البعض الآخر عن التطور كما هو الحال في تقنيات أنصاف النواقل. سنستعرض في هذا الجزء مبدأ تنفيذ المؤثرات المنطقية باستخدام الأزرار الكهربائية ثم أنصاف النواقل دون الدخول في تفاصيل (هامة من حيث المبدأ) يمكن لمعرفتها العودة إلى كتب مختصة.

سنصطلح في كل ما سيأتي، على تمثيل الواحد المنطقي بالقيمة العليا للجهد الكهربائي V^+ (5 فولت مثلاً) والصفر المنطقي بالقيمة الدنيا للجهد الكهربائي V^- (0 فولت مثلاً).

2 تنفيذ المؤثرات المنطقية بتقنية الأزرار

وهي أقدم التقنيات وأبسطها. تتألف الأزرار من نابض حامل لقطعة معدنية مُوصِلة كهربائياً ومغلّفة بعازل، يؤدي الضغط عليها لوصل النقطتين A، B المنفصلتين في حالة الراحة (أي عدم وجود أي

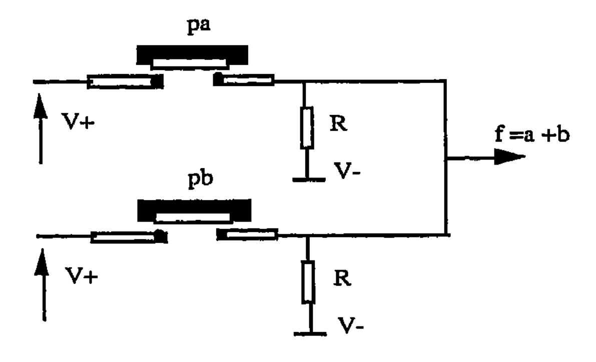
تأثير على النابض) كما في الشكل:



فعند ضغط الزر p تتصل النقطة A بالنقطة B، ونجد الجهد V^+ عند النقطة B وإلا نجد القيمة V^-. أي نجد عند B الواحد عند الضغط على الزر والصفر عند تركها في حالة راحة.

البوابة OR:

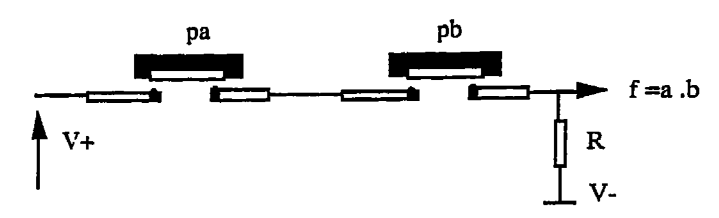
لنلحق بالمتغيرين المنطقيين a و b زرين pa و pb ولنصلهما على التوازي:



فعند الضغط على أي من الزرين أو على كليهما نجد عند الخرج الجهد V^+ (أي الواحد)، وإلا نجد القيمة V^- (أي الصفر).

البوابة AND:

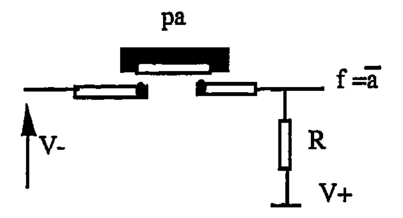
هي كما في المؤثر OR ولكن الوصل يكون على التسلسل:



حيث نجد عند الخرج f القيمة V^- (أي الصفر) إلا إذا ضُغط على الزرين معاً، فنجد عندها القيمة V^+ (أي الواحد).

البوابة NOT:

يمكن تحقيق هذا المؤثر كما في الشكل التالي:



حيث نجد القيمة V^+ عند f إلا إذا ضغطنا على الزر، فنجد عندها القيمة V^-عند f.

تشكل البوابات الأخرى بتراكيب مختلفة من البوابات المذكورة آنفاً.

3 تنفيذ المؤثرات المنطقية بأنصاف النواقل

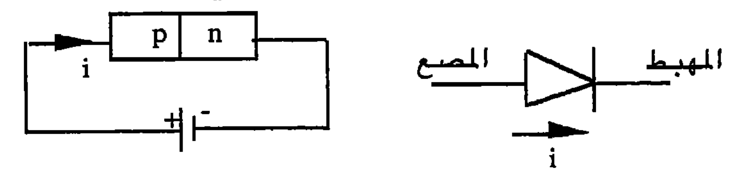
تطورت تقنية تحقيق المؤثرات المنطقية باستخدام أنصاف النواقل تطوراً كبيراً، وجرى، جيلاً بعد جيل، تعزيز المواصفات الإلكترونية لهذه المؤثرات من حيث الاستطاعة المستهلكة والتأثر بالضجيج ودرجة الحرارة وتردد العمل، الخ... استخدمت في بادئ الأمر الديودات والمقاومات لتنفيذ المؤثرين المنطقيين AND و OR ، ثم استخدمت الترانزستورات في بنى وتقنيات مختلفة.

سنقدم فيما يلي فكرة مبسطة عن الديود والترانزستور

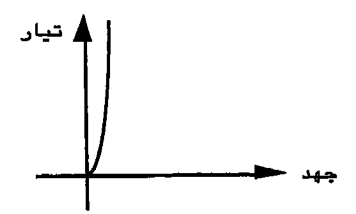
الديود:

هو وصلة نصف ناقل (سيليسوم مثلاً) جرى تطعيم الجزء الأول منها بالفوسفور أو الزرنيخ، وبذلك يصبح هذا الجزء حاملاً للإلكترونات (n)، في حين جرى تطعيم الجزء الثاني بالغاليوم أو الأنديوم، وبذلك يصبح حاملاً للثقوب (p).

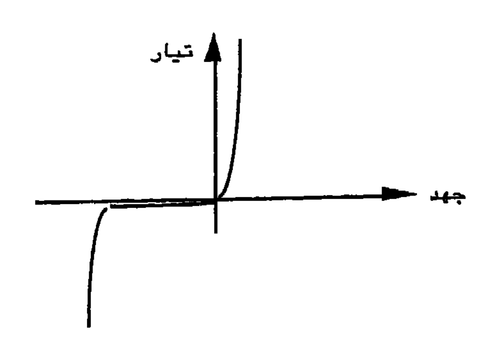
عند ربط الديود بدارة كما في الشكل التالي:



تتحرك الإلكترونات باتجاه الثقوب ويسري تيار نسميه بتيار الاستقطاب المباشر، ويكون لمميز التيار-الجهد الشكل العام التالي:



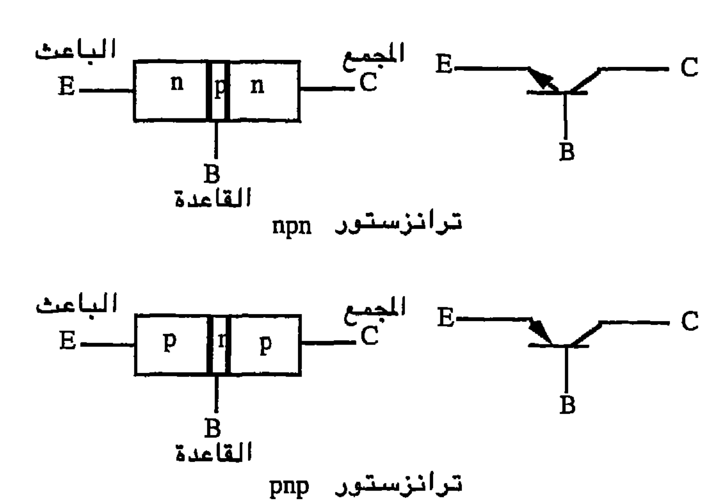
أما في حالة الاستقطاب المعاكس فيسري تيار ضعيف جداً (نظرياً معدوم)، إلا إذا تجاوز الجهد قيمة محددة فعندها يتزايد التيار نتيجة ما نسميه بانهيار الديود. وبالتالي سيكون للميز العام تيار-جهد الشكل التالي:



والذي يظهر أن الديود يكون ممرراً للتيار عند تطبيق جهد صغير (غير معدوم وتتوقف قيمته على مواصفات الديود)، وتكون مقاومته شبه معدومة. ولكن عندما يكون الجهد سالباً بين المصعد والمهبط، يكون التيار شبه معدوم (مع تغيير في جهته) أي إن المقاومة تصبح كبيرة جداً. وعندما يزيد هذا الجهد عن مقدار معين يزداد التيار العكسي وينهار الديود (وهو ما نتجنب حدوثه عامة إلا في بعض التطبيقات الخاصة).

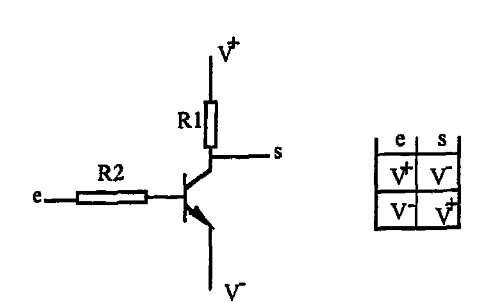
الترانزستور ثنائي القطبية:

تتألف وصلة الترانزستور من شريحة من السيليسيوم جرى تطعيم طرفيها اليميني واليساري بالفوسفور مثلاً للحصول على منطقتين حاملتين للإلكترونات (n)، في حين جرى تطعيم منطقة ضيقة جداً بالغاليوم للحصول على منطقة حاملة للثقوب (p). تنشأ بذلك وصلة الترانزستور npn. ويمكن بطريقة معاكسة الحصول على ترانزستور من نوع pnp. يمثّل كل نوع من الترانزستورات كما يلي:



تسمى مناطق الترانزستور الثلاث بالباعث Emitter والقاعدة Base والمجمع Collector. ويشير سهم الباعث إلى اتجاه التيار بين الباعث والمجمع، وذلك عند تطبيق جهد مناسب بين القاعدة والباعث. يكون هذا التيار أعظمياً عند تطبيق جهد موجب، أكبر من قيمة محددة نسميها جهد الإشباع، بين القاعدة والباعث، في حالة الترانزستورات npn. ويكون معدوماً عند تطبيق جهد سالب أو معدوم بين القاعدة والباعث. أي إن المقاومة بين المجمع والباعث تكون نظرياً معدومة عند تطبيق جهد موجب على القاعدة بالنسبة إلى الباعث، وتكون هذه المقاومة لانهائية نظرياً عند تطبيق جهد معدوم أو سالب على القاعدة بالنسبة إلى الباعث. ويمكن قول الشيء نفسه فيما يتعلق بالترانزستورات pnp، ولكن بعد عكس جهة الجهد المطبق على القاعدة بالنسبة إلى الباعث. تقوم القاعدة إذن بالتحكم بمرور أو قطع التيار بين المجمع والباعث. تسمى هذ الترانزستورات بالثنائية الاستقطاب BJT (Bipolar Jonction Transistor).

يستخدم الترانزستور لتحقيق البوابة NOT وفق المخطط التالي:

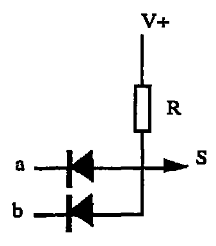


e	s
V^+	V^-
V^-	V^+

1-3 تقنية الديود-الديود

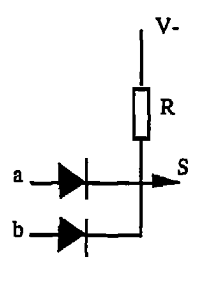
DDL (Diode-Diode Logic)

البوابة AND:



نجد عند النقطة S القيمة V^+ فقط عندما يكون لكلا المتغيرين a و b القيمة V^+. أما إذا كان لأي منهما أو لكليهما القيمة V^- نجد القيمة V^- عند الخرج S. وهذا هو عمل البوابة AND.

البوابة OR:

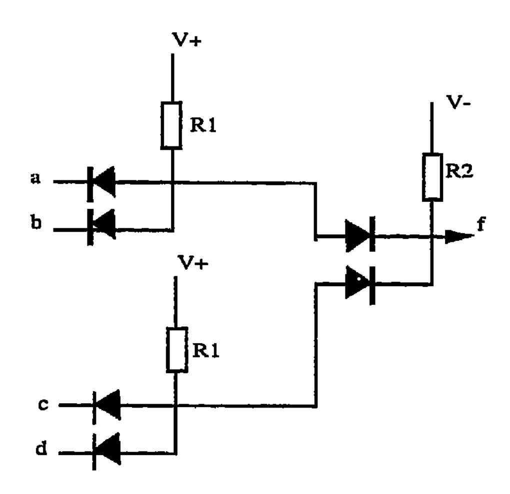


نجد عند النقطة S القيمة V^+ عندما يكون لأي من المتغيرين a أو b أو لكليهما القيمة V^+. ويكون S مساوياً لـ V^- إذا كان لكل من a و b القيمة نفسها V^-.

وبالرغم من سهولة تنفيذ هذه البوابات، فإن لها بعض المشاكل التي تعوق استخدامها. مثلاً، ليكن مطلوباً تحقيق الدالة المنطقية ذات المتغيرات الأربعة a، b، c، d التي صيغتها:

f (a,b,c,d) = ab + cd

باستخدام بوابات ذات مدخلين. يمكن تحقيق ذلك بدارة لها المخطط التالي:



فحين يكون كل من a وb و c وd القيمة V^+، نجد أن الجهد عند النقطة f، من أجل $V^- = 0$، هو:

$$V_f = \frac{V^+}{\frac{R_1}{2}+R_2} R_2$$

أما إذا طبق على المدخل c الجهد $V^- = 0$، فنجد عندئذ أن جهد النقطة f يساوي:

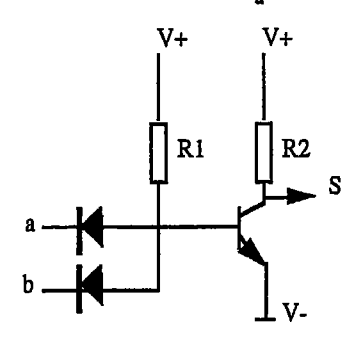
$$V^-_f = \frac{V^+}{R_1 + R_2} R_2$$

أي أن $\overline{V}f$ يختلف بوضوح عن Vf، وهذا غير مقبول، إذ يجب أن نحصل على الجهد نفسه في كلتا الحالتين، ذلك أن للدالة f القيمة المنطقية نفسها. ولقد دفع هذا النمط من المشاكل إلى البحث عن تطوير تقنيات أخرى لا نواجه فيها مثل هذه المصاعب.

2-3 تقنية الديود-الترانزستور

DTL (Diode-Transistor Logic)

تتمثل المشكلة كما رأينا سابقاً في الخرج، الذي يتغير الجهد عنده بدلالة الدخل، وذلك بالرغم من وجوب محافظة الخرج على القيمة المنطقية نفسها. ولمواجهة هذه المشكلة يمكن أن نستخدم الترانزستور كمبدل بين V^- و V^+ (أي تشغيله بالإشباع) لتنفيذ بوابة NAND (وهي مؤثر تام) وفق المخطط التالي:

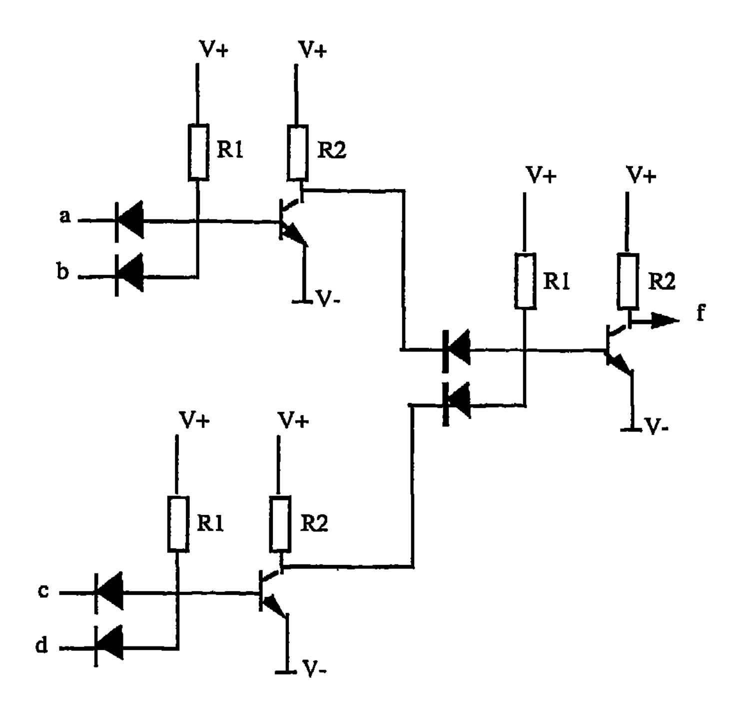


حيث تحافظ S على القيمة V^+ إلا إذا كان لكل من a و b الجهد V^+.

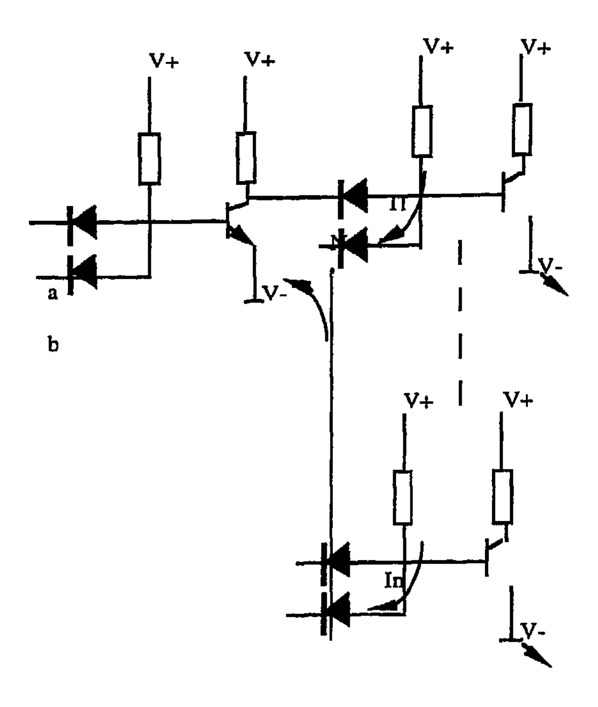
ويمكننا تحقيق دالة كالتي رأيناها سابقاً $f = ab + cd$ بعد ملاحظة أن:

$$f = \overline{\overline{ab + cd}} = \overline{\overline{ab} . \overline{cd}}$$

ومنه المخطط التالي:



ولما كان الخرج في تقنية الـ DTL هو ترانزستور، فإنه يخلصنا من مشاكل الـ DDL (من نمط تلك التي تحدثنا عنها سابقاً). ولكن هذه التقنية لا تخلو بدورها من نقاط ضعف، مثل عدد البوابات التي يمكن وصلها إلى بوابة وحيدة، أو ما نسميه Fan Out. ولنوضح ذلك كما يلي:



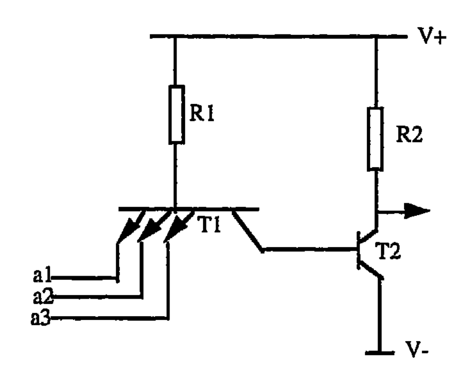
فلو قمنا بوصل عدة بوابات NAND إلى خرج بوابة NAND أولى، كما في الشكل السابق، ولو كان خرج هذه البوابة مساوياً الصفر، لوجب عندها مرور تيارات من بوابات المرحلة الثانية تسبب ارتفاع الجهد عند النقطة N. ذلك لأن الترانزستور لن يسمح بمرور تيار أكبر من قيمة محددة بين المجمع والباعث، وهذا ما يفرض أن يكون عدد بوابات المرحلة الثانية التي يمكن وصلها إلى بوابة المرحلة الأولى محدوداً.

3-3 تقنية الترانزستور-الترانزستور

TTL (Transistor-Transistor Logic)

بنيت الدارات المتكاملة الأولى على أساس تقنية الـ DTL التي تطورت بعد ذلك إلى تقنية الـ TTL. ذلك أن عملية تصنيع دارات مبنية فقط على المقاومات والترانزستورات أسهل من الدارات المبنية على المقاومات والترانزستورات والديودات. يستخدم في

تقنية الـ TTL ترانزستورات متعددة البواعث. وسنوضح عمل الدارات من هذه التقنية بواسطة مثال: بوابة NAND بثلاثة مداخل:



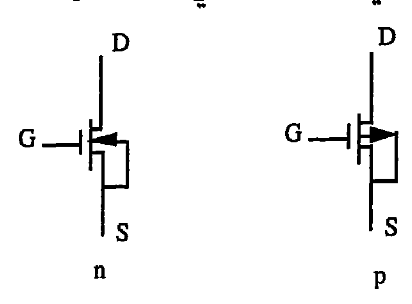
نلاحظ أن مجمع الترانستور T_1 متصل بباعثه دائماً. فإذا كانت قيمة أي من المداخل الثلاثة a_1 أو a_2 أو a_3 هي الصفر، بقي مجمع هذا الترانزستور على الجهد V^-، ومن ثم فالخرج S سيكون مساوياً لـ V^+، أي الواحد. ولكن عندما تأخذ المداخل الثلاثة a_1 و a_2 و a_3 معاً القيمة V^+ (أي الواحد) يأخذ مجمع الترانزستور T_1 هذه القيمة أيضاً، ومن ثم يصبح الترانزستور T_2 ممرراً وتكون قيمة S مساوية لـ V^- (أي للصفر)، وهذا هو عمل بوابة NAND.

4-3 تقنية المعدن-الأكسيد-نصف الناقل

تبين الدارة المعروضة آنفاً مبدأ دارة NAND بتقنية TTL. وتحتاج هذه الدارة إلى تعديلات كثيرة لكي تستجيب لمتطلبات عديدة، مثل زمن الاستجابة والاستطاعة المستهلكة، التي سنتحدث عنها لاحقاً. (للاستزادة، يمكن العودة إلى كتب الالكترونيات.)

استخدمت في الدارات المنطقية المتكاملة تقنية هامة أخرى هي تقنية MOS (Metal Oxide Semiconductor) (أو المعدن-الأكسيد-نصف الناقل) التي تبنى على نوع مختلف من

الترانزستورات هو ترانزستور أثر الحقل FET (Field-Effect Transistor) بنوعيه: ذي القنال n وذي القنال p.



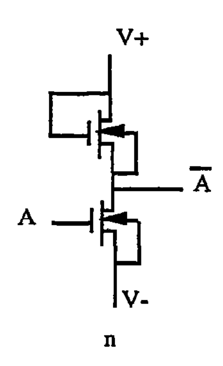
يمكن، في هذا النوع من الترانزستورات، التحكم في الممانعة بين المنبع Source والمصرف Drain وذلك بالتأثير على الزالقة Gate، بفضل حقل كهربائي تولده الشحنة الساكنة المطبقة على الزالقة (من هنا جاءت التسمية: أثر الحقل). ففي الترانزسيتور من النمط n مثلاً، يكوّن الخط من المصرف إلى المنبع دارة مفتوحة (مقاومة لانهائية) عندما يكون جهد الزالقة سالباً بالنسبة إلى المنبع. أما إذا كان جهد الزالقة موجباً بالنسبة إلى المنبع، فيكون خط الدارة بين المصرف والمنبع مقصوراً (مقاومة معدومة). ويكون الأمر على عكس ذلك في حالة ترانزيستورات الحقل من النمط p.

نلاحظ هنا أنه لا يوجد تيار بين الزالقة والمنبع، وهذا هو الاختلاف الأول مع الترانزستور ذي الوصلة ثنائية الاستقطاب BJT، حيث يجري التحكم في قيمة هبوط الجهد بين المجمع والباعث بالتيار الذي يعبر القاعدة باتجاه الباعث، أما الاختلاف الثاني فيكمن في أن ترانستور BJT، عندما يكون ممرراً ومشبعاً، يبقى فيه هبوط جهد صغير بين المجمع والباعث، وهذا يعني استهلاكاً للطاقة. أما في ترانزستورات الحقل، فعندما يكون الترانزستور ممرراً تكون الممانعة بين المنبع والمصرف معدومة. يؤدي كل هذا إلى كون الطاقة المستهلكة في ترانزستورات الحقل صغيرة جداً مقارنة بالطاقة المستهلكة في الترانزستورات الثنائية الاستقطاب.

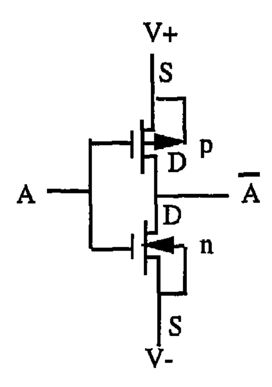
الحامل في ترانزستورات n-MOS هو الإلكترونات، وهو الثقوب

Holes في ترانزستورات p-MOS. ولما كانت حركة الإلكترونات أسرع من حركة الثقوب مانت ترانزستورات n-MOS أسرع من p-MOS.

يظهر الشكل التالي مخطط بوابة العاكس باستخدام ترانزستورات من نوع n-MOS:



يُستخدم في تقنية CMOS (Complementary MOS) كلا النوعين من الترانزستورات. ويكون لبوابة العاكس المخطط التالي:



تمتاز تقنية CMOS عن تقنية nMOS بكونها ذات استهلاك أقل وممانعة خرج أصغر.

4 خواص ومواصفات البوابات المنطقية

يجب الانتباه عند استخدام البوابات المنطقية إلى نقاط عديدة،

مثل زمن الانتشار وهامش الضجيج والاستطاعة التي تستهلكها البوابة الواحدة... تحدد هذه المواصفات حدود التعامل مع البوابات المنطقية الإلكترونية كما ستوضح ذلك الفقرات التالية.

1-4 زمن الانتشار

إن الزمن اللازم لانتقال بوابة منطقية من حالة منطقية إلى أخرى ليس معدوماً، فزمن التبديل في الترانزستور، وهو الزمن اللازم لكي يقوم تيار القاعدة بفتحه أو إغلاقه، ليس معدوماً نتيجة للأثر السعوي بين القاعدة والباعث، وكذلك بين المجمع والباعث. يؤدي هذا إلى حدوث تأخير بين لحظة تطبيق الإشارة عند دخل البوابة ولحظة التغير المنشود في الخرج. نسمي هذا التأخير بزمن الانتشار tp الذي تعتمد قيمته على الدارة والعناصر المستخدمة في تحقيق البوابة. يحدد هذا التأخيرتردد الإشارة الأعظمي التي يمكن تطبيقها على دخل البوابة دون أن يصيبها تشوه كبير، ويعطى بالعلاقة:

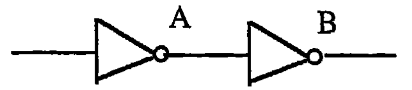
$$Fmax = \frac{1}{2\, t_p}$$

2-4 مستوى إشارة الدخل والخرج

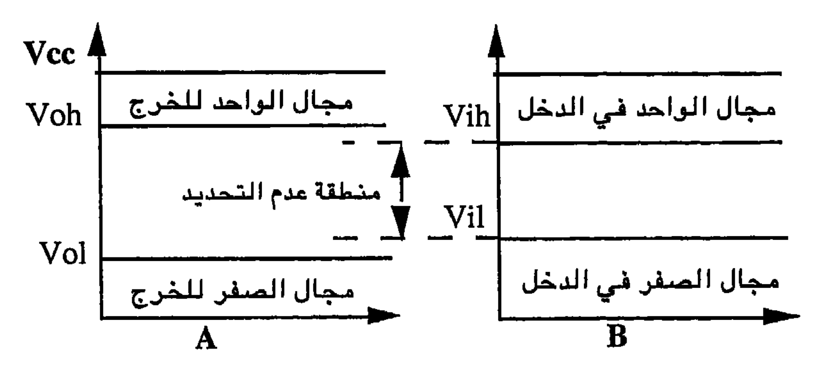
تحدثنا حتى الآن عن جهد كهربائي مقابل للصفر ورمزنا إليه بـ V^- وجهد آخر مقابل للواحد رمزنا إليه بـ V^+. ولكننا ندرك بسهولة أن جعل الصفر مساوياً لقيمة محددة تماماً V^- يقود إلى العديد من المشاكل، وكذلك الأمر بالنسبة للواحد وقرنه بالقيمة V^+. فالضجيج الكهربائي الذي يتراكم على إشارتي الدخل والخرج يغير من مستوياتهما، وكذلك تغير جهد الخرج، الذي هو دخل لبوابة منطقية تالية، بسبب تغير الحمل...

لذا يجب تحديد مستوى الصفر ومستوى الواحد على شكل مجال.

ولكن هل يجب أن يكون مجال الصفر مثلاً هو نفسه عند الدخل والخرج؟ الجواب: لا، فعلى مجال الخرج أن يكون أصغر من مجال الدخل، ذلك أن خرج بوابة A قد يوصل بدخل أخرى B كما في الشكل:



وعلى البوابة الثانية ألا تخطىء في تعاملها مع الصفر الذي يظهر على البوابة الأولى، بالرغم مما قد يتراكب مع خرج هذه البوابة من ضجيج. أما الواحد، فيجب أن يكون في بوابة الخرج A أعلى منه في B. وهذا ما يمثّله المخططان التاليان:



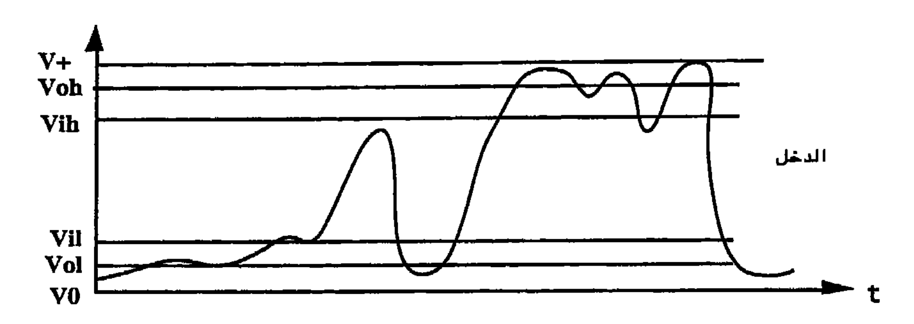
حيث يشير Vil إلى أكبر قيمة لجهد في الدخل تقبله البوابة المنطقية كصفر. أما Vol فتشير إلى أكبر قيمة لجهد تعطيه البوابة كصفر في خرجها. وأما Vih فهو أصغر جهد تقبله البوابة المنطقية على أنه مساو للواحد. وأما Voh فهو أصغر خرج تعطيه بوابة منطقية كواحد على خرجها.

نسمي الفارق Voh-Vih بمناعة الواحد المضادة للضجيج (هامش الضجيج للواحد)، ذلك أن أي انخفاض في قيمة جهد خرج بوابة منطقية ما بالمقدار Voh-Vih، بسبب الضجيج الخارجي، لن يؤثر في قبول بوابة منطقية أخرى للقيمة Vih كواحد.

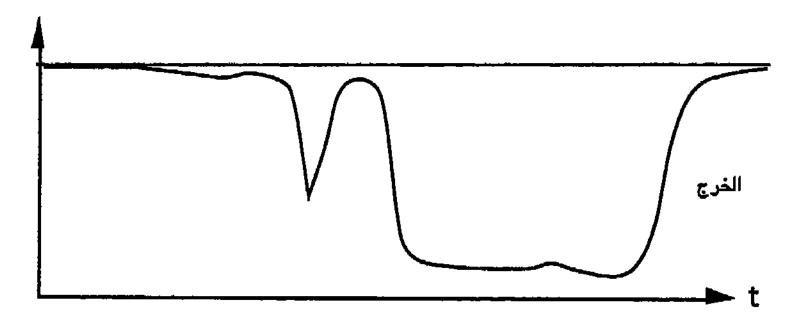
ونسمي الفارق Vil -Vol بمناعة الصفر المضادة للضجيج (هامش الضجيج للصفر). ذلك أن أي ارتفاع في قيمة جهد خرج بوابة منطقية

ما بالمقدار Vil -Vol، بسبب الضجيج الخارجي، لن يؤثر في قبول بوابة منطقية أخرى للقيمة Vil كصفر.

يوضح المخطط التالي تأثير ذلك في إشارة دخل يتأرجح الجهد فيها بين قيم مختلفة:



والناتج (الخرج) في حالة بوابة عاكس:



لنلاحظ أن هذه المناعة هي سكونية (ستاتيكية)، وهي أكبر من ذلك في حالة الإشارة العابرة السريعة التي تدوم زمناً أقصر من زمن الانتشار في البوابات.

3-4 جهد التغذية

يعتمد جهد التغذية $(V^+ - V^-)$ على العائلة المستخدمة من TTL أو MOS، الخ... قيمة جهد التغذية الاسمي يساوي عموماً 5V، وذلك في لتقنيات التي تعتمد على الترانزستورات الثنائية القطبية BJT

التي تعمل في الإشباع. أما عائلة MOS فيمكن أن يصل جهد التغذية فيها إلى 15V.

4-4 الاستطاعة المبددة

وهي الاستطاعة التي تستهلكها البوابة المنطقية. هذه الاستطاعة متغيرة تبعاً لكون البوابة المنطقية في حالة واحد أو في حالة صفر؛ وتعطى عادة كقيمة وسطى، أي وسطي الاستطاعة المبددة لبوابة في حالة الواحد والصفر. وعلى العموم، تتعلق الاستطاعة المبددة بالعوامل التالية

- جهد التغذية؛
- هوامش الضجيج؛
- تردد العمل؛
- الشحن السعوية.

تتغير الاستطاعة المبددة تبعاً للبوابة المنطقية والتقنية المستخدمة في تنفيذها، وتتغير بين 1mW و 50 mW.

5-4 درجة حرارة الوسط

تتأثر مكونات أنصاف النواقل بدرجة حرارة الوسط، إذ لايمكن لبوابة منطقية أن تعمل عملاً صحيحاً إلا ضمن مجال درجة حرارة محدد يتعلق بنوع التطبيقات التي تستخدم فيها هذه البوابات.

تتوافر البوابات المنطقية في ثلاث فئات هي:

- الفئة العسكرية : ومجال عملها يقع بين ^{0}C 55- و ^{0}C 125.
- الفئة الصناعية : ومجال عملها يقع بين ^{0}C 25- و ^{0}C 85.
- الفئة المدنية : ومجال عملها يقع بين ^{0}C 0- و ^{0}C 85.

5 الدارات المتكاملة

يمكن جمع عدة بوابات منطقية في دارة واحدة لها التغذية نفسها، لتكوّن ما نسميه بالدارة المتكاملة IC (Integrated Circuit). نسمي الدارات المتكاملة التي لا يزيد عدد البوابات فيها عن عشر بالدارات الصغيرة التكامل SSI (Small Scale Integration)، أما الدارات المتكاملة التي تضم أكثر من عشر بوابات وأقل من مئة بوابة فنسميها بالدارات المتوسطة التكامل MSI (Medium Scale Integration)، وأما الدارات التي تضم أكثر من مئة بوابة فنسميها بالدارات العالية التكامل LSI (Large Scale Integration). وأخيراً، نسمي الدارات التي تضم آلاف البوابات بالدارات العالية التكامل جداً VLSI (Very Large Scale Integration).

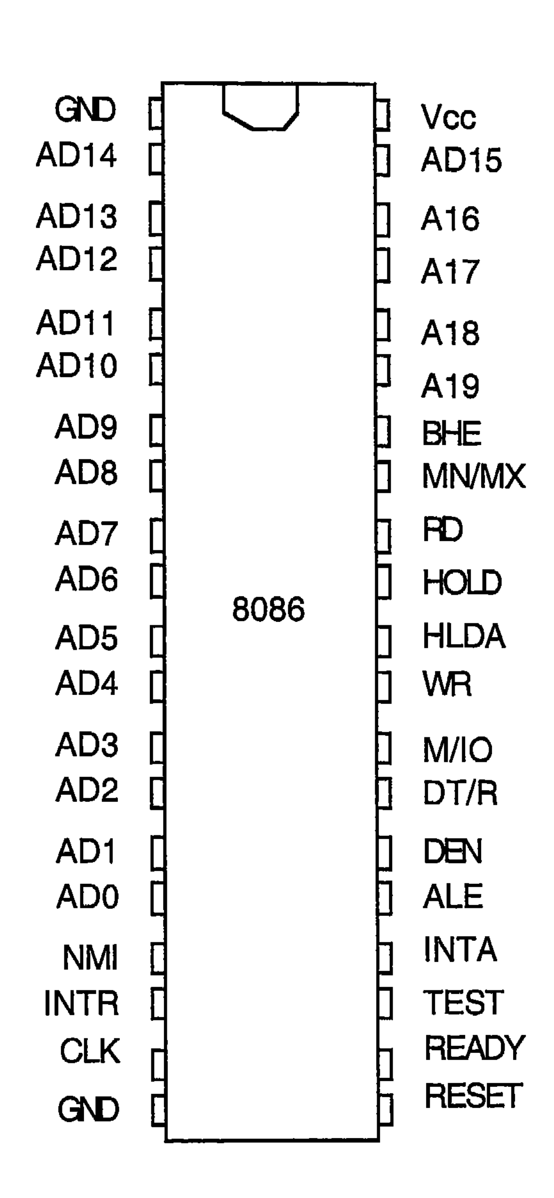


الملحق الثالث

وصف مرابط المعالج 8086

عرضنا في الفصل الأول من الجزء الأول من هذا الكتاب المعالج الصغري 8086 من وجهة نظر برمجية، فذكرنا بنيته الداخلية، وسجلاته، وعدد خطوط العنونة، وخطوط المعطيات. أي ذكرنا ما يحتاج إليه المستثمر لبرمجة المعالج. ونعرض في هذا الملحق المعالج الصغري من وجهة نظر إلكترونية، فنصف مرابطة الخارجية Pin وعملها.

للمعالج 8086 أربعون مربطاً خارجياً (انظر الشكل 1) نعطي فيما يلي وصفاً لوظائفها.



الشكل 1: المرابط الخارجية للمعالج الصغري 8086.

اسم المربط	رقمه	نوعه	وظيفته
AD_0 - AD_{15}	من 2 إلى 16 و39	د/خ	مسرى العناوين والمعطيات. يضع المعالج على هذه الخطوط أولا العنوان المراد الوصول إليه، ثم يضع القيمة المراد كتابتها في ذلك العنوان (في حالة الكتابة)، أو يقرأ المعطيات التي تضعها الذاكرة على هذه الخطوط (في حالة القراءة).
S6/A19 S5/A18 S4/A17 S3/A16	من 35 إلى 38	خ	خطوط عنونة/حالة. تمثل هذه الخطوط الأوزان العليا للعنوان في مرحلة وضع العناوين على المسرى. ثم تنقل بعد ذلك معلومات عن حالة المعالج في المراحل الأخرى، فمثلاً تدل هذه الخطوط على سجل القطاع المستخدم حالياً في المعالج.
S7/$\overline{BHE}$	34	خ	يؤهل المعالج هذا الخط عندما يرغب في الوصول إلى الجزء العلوي من مسرى المعطيات (D_9 إلى D_{15}).
$\overline{RD}$	32	خ	يدل هذا الخط على قيام المعالج بعملية قراءة من موقع في الذاكرة أو معبر د/خ.
READY	22	د	يسمح هذا المدخل للمعالج بمعرفة جاهزية الذاكرة أو معبر د/خ لإتمام تبادل المعلومات.
INTR	18	د	يفحص المعالج هذا المدخل دورياً (عند كل تعليمة) لتحسس وجود طلب مقاطعة من طرفية خارجية.

$\overline{TEST}$	23	د	يفحص المعالج هذا الدخل عند تنفيذه لتعليمة wait،فإذا وجد القيمة '0' فإنه سينتقل إلى التعليمة اللاحقة وإلا فسيدخل في حالة ركود إلى أن تظهر القيمة '1' أو يعاد إقلاعه.
NMI	17	د	المقاطعة غير القابلة للحجب. يؤدي ظهور جبهة على هذا الخط إلى مقاطعة عمل المعالج. تتميز هذه المقاطعة بأنها لا تحجب برمجياً، فظهور إشارة على هذا المدخل سيؤدي حتماً إلى مقاطعة المعالج.
RESET	21	د	مدخل الاستهلال. عندما يأخذ هذا المدخل القيمة '1' فإن المعالج يتوقف عن العمل وبعودة هذا المدخل إلى '0' يقلع المعالج من جديد، وينفذ برنامجه من البداية.
CLK	19	د	دخل الساعة. ينبغي تطبيق إشارة مربعة دورية على هذا الدخل لتقوم بدور ساعة العمل التي توقّت عمل المعالج.
VCC	40		تغذية. يُربط هذا الدخل إلى الجهد المستمر +5V لتغذية المعالج.
GND	1,20		أرضي. يربط هذان المربطان إلى المأخذ الأرضي (0V).
MN/$\overline{MX}$	33	د	تحدد القيمة على هذا الدخل نمط العمل. فللمعالج نمطا عمل: أصغري يُستخدم في النظم الوحيدة المعالج، وأعظمي يسمح للمعالج بالتعايش مع معالجات أخرى.

أما المرابط المتبقية فيختلف دورها تبعاً لنمط العمل. ففي النمط الأصغري يكون عمل المرابط كما يلي:

اسم المربط	رقمه	نوعه	وظيفته
$M/\overline{IO}$	28	خ	يُستخدم للتمييز بين قيام المعالج بالوصول إلى ذاكرة أو إلى د/خ.
$\overline{WR}$	29	خ	تدل هذه الإشارة، عندما تصبح فعّالة، أن المعالج يريد الكتابة في أحد مواقع الذاكرة أو في أحد المعابر.
$\overline{INTA}$	24	خ	إشعار بقبول المقاطعة. يؤهل المعالج هذا الخرج عندما يقبل طلب المقاطعة INTR، فيشعر بذلك الطرفيات المعنية بقبوله الطلب.
ALE	25	خ	تأهيل لاقف العناوين. يتأهل هذا الخرج عندما يضع المعالج عنواناً على مسراه، ويعود هذا الخرج إلى القيمة 0 عندما ينتقل المعالج إلى مراحل عمل أخرى.
$DT/\overline{R}$	27	خ	إرسال معطيات أو استقبالها. يفيد هذا الخرج في تحديد جهة انتقال المعلومات من المعالج نحو الطرفيات أو بالعكس.
$\overline{DEN}$	26	خ	تأهيل المعطيات. تصبح هذه الإشارة فعّالة عندما ينتقل المعالج من مرحلة العنونة إلى مرحلة وضع أو قراءة المعطيات.

HOLD	30	د	طلب التجمد. قد تطلب دارة خارجية إلى المعالج التوقف عن العمل، فتعبر تلك الدارة عن طلبها بواسطة هذا الدخل.
HLDA	31	خ	قبول التجمد. يعرب المعالج عن قبول طلب التجمد HOLD بتأهيل هذا الخرج. وعند قبوله للتجمد فإنه يرفع سيطرته عن مسرى المعطيات والعناوين والتحكم. يخرج المعالج من حالة التجمد عندما تزول إشارة طلب التجمد HOLD.

وفي النمط الأعظمي، تأخذ المرابط السابقة الدلالات التالية:

اسم المربط	رقمه	نوعه	وظيفته
$\overline{S0}$, $\overline{S1}$, $\overline{S2}$	28–26	خ	حالة المعالج. تدل هذه المخارج على العملية التي يقوم المعالج بتنفيذها (قراءة/كتابة/ركود).
$\overline{RQ_0}/\overline{GT_0}$ $\overline{RQ_1}/\overline{GT_1}$	30 31	د/خ	طلب المسرى/إشعار بالإخلاء. عندما يرغب معالج آخر في الوصول إلى الذاكرة فإنه يطلب من المعالج الحالي إخلاء المسرى. بعد قبول طلب الإخلاء، يرسل المعالج إشعاره بالقبول على الخط ذاته.
$\overline{LOCK}$	29	خ	إقفال المسرى. يدل تأهيل هذا الخرج على أن المعالج يحجز جميع الموارد لمصلحته فلا يسمح لغيره بالوصول إليها.
QS_0 , QS_1	25–24	خ	حالة الرتل. يدل هذان المخرجان على حالة رتل المعالج.

الملحق الرابع

مجموعة تعليمات المعالج 8086

يمكن أن نقسم تعليمات المعالج 8086 إلى ست فئات: تعليمات نقل المعطيات، والتعليمات الحسابية، والتعليمات المنطقية، وتعليمات سلاسل المحارف، وتعليمات القفز، وتعليمات التحكم في المعالج.

1 تعليمات نقل المعطيات

تُصنف تعليمات نقل المعطيات Data Transfer Instructions بدورها في مجموعات فرعية تبعاً لعملها. وسنعرض تباعاً هذه المجموعات.

1-1 تعليمات النقل لثمانيّة أو لكلمة

- رمز التعليمة: MOV dst, src

وصف التعليمة:

نقل كلمة أو ثمانيّة من المصدر src إلى الوجهة dst.

أنماط العنونة:

– فورية. مثال:

MOV CX, 037Ah

– مباشرة. مثال:

MOV BL, [437Ah]

- بواسطة السجل. مثال:

MOV BX, AX

- غير مباشرة بالسجل. مثال:

MOV DL, [BX]

• رمز التعليمة: PUSH src16

وصف التعليمة:

نقل الكلمة المحددة بـ src16 إلى أعلى المكدس. بعد تنفيذها، يُنقص مؤشر المكدس بمقدار موقعين.

أنماط العنونة:

- بواسطة السجل. مثال:

PUSH BX

- مباشرة. مثال:

PUSH table[BX]

ينقل المعالج القيمة ذات الانزياح table + [BX] إلى المكدس.

• رمز التعليمة: POP dst16

وصف التعليمة:

نقل الكلمة المرمزة على 16 خانة إلى أحد سجلات المعالج أو إلى موقع في الذاكرة

أنماط العنونة:

- بواسطة السجل. مثال:

POP DX

- غير مباشرة بالسجل. مثال:

POP table[BX]

• رمز التعليمة: XCHG dst, src

وصف التعليمة:

تبادل محتوى المصدر والوجهة، وهي تتعامل مع الثمانيّات والكلمات.

أنماط العنونة:

– بواسطة السجل. مثال:

```
XCHG AX, DX    ; instruction operating on 16 bits
XCHG AL, CH    ; instruction operating on 8 bits
```

– مباشرة. مثال:

```
XCHG AL, Prices [Bx]
```

يبادل المعالج بين السجل AL وموقع الذاكرة ذي الانزياح Prices + Bx.

• رمز التعليمة: XLAT

وصف التعليمة:

استبدال بمحتوى السجل AL ثمانيّة مقروءة من جدول تقابل مخزّن في الذاكرة LUT. يجب، قبل تنفيذ هذه التعليمة، تخزين جدول التقابل في الذاكرة، وتخزين انزياح عنوان الجدول الابتدائي في السجل BX. بعد ذلك، عند تنفيذ التعليمة، فإنها تنقل محتوى الذاكرة ذات الانزياح BX+AL إلى السجل AL.

أنماط العنونة:

– بواسطة السجل (في الواقع التعليمة لا تقبل حدوداً). مثال:

```
MOV BX, 2800h        ; top of table in BX
XLAT                 ; replace ASCII in AL
```

2-1 تعليمات الدخل والخرج

• رمز التعليمة: IN ACC, src

وصف التعليمة:

قراءة كلمة أو ثمانيّة من المعبر، ووضعها في المراكم.

أنماط العنونة:

– فورية. مثال:

```
IN AL, 0C8h
```

نقل ثمانيّة من المعبر 0C8h إلى السجل AL.

IN AX, 34h

نقل كلمة مرمزة على 16 خانة من المعبر 34h إلى السجل AX.

- رمز التعليمة: OUT dstport, ACC

وصف التعليمة: نقل ثمانيّة أو كلمة من المراكم إلى المعبر.

أنماط العنونة:

– فورية. مثال:

```
OUT 3Bh, AL      ; instruction operating on 8 bits
OUT 2Ch, AX      ; instruction operating on 16 bits
```

3-1 تعليمات خاصة

- رمز التعليمة: LEA register, src

وصف التعليمة:

تحدد هذه التعليمة انزياح المصدر src (الذي قد يكون متحولاً أو موقع ذاكرة)، وتضع قيمة هذا الانزياح في السجل المذكور (ذي 16 خانة).

أنماط العنونة:

– بواسطة السجل. مثال:

LEA BX, Prices

يضع المعالج انزياح الموقع Prices في السجل BX.

LEA BP, SS:StackTop

يضع المعالج في BP انزياح الموقع StackTop الموجود في قطاع المكدس.

– غير مباشرة بواسطة السجل. مثال:

LEA CX, [BX][DI]

ينقل المعالج إلى CX الانزياح الناتج من جمع محتوى السجلين BX و DI.

• رمز التعليمة: LDS register, memory adr

وصف التعليمة:

تشحن هذه التعليمة سجلاً ذا 16 خانة بمحتوى موقع الذاكرة المحدد في التعليمة، والموقع التالي له، ثم تشحن السجل DS بقيمة موقعي الذاكرة التاليين.

تفيد هذه التعليمة في شحن السجلين DS و SI ليؤشرا على بداية سلسلة محارف معينة قبل استخدام التعليمات الخاصة بسلاسل المحارف.

أنماط العنونة:

– مباشرة. مثال:

LDS BX, [4326h]

تنسخ التعليمة محتوى الذاكرة ذا الانزياح 4326hإلى السجل BL، ومحتوى الموقع 4327h إلى السجل BH، ومحتوى الموقعين 4328h و 4329h إلى السجل DS.

LDS SI, string pointer

يشحن المعالج السجل SI بمحتوى الموقعين string pointer و string pointer+1، ويشحن السجل DS بمحتوى الموقعين string pointer+2 و string pointer+3.

• رمز التعليمة: LES register, mem-adr

وصف التعليمة:

تنقل هذه التعليمة قيمة موقعين متتاليين من الذاكرة داخل السجل المحدد (ذي 16 خانة)، ثم تشحن داخل السجل ES قيمة الموقعين التاليين للذاكرة.

تفيد هذه التعليمة، مثلاً، في شحن المؤشرين DI وES، ليؤشرا على بداية سلسلة المحارف قبل استخدام التعليمات الخاصة بهذه السلسلة.

أنماط العنونة:

- مباشرة. مثال:

LES BX, [789Ah]

تشحن التعليمة محتوى الموقعين 789Ah و 789Bh داخل السجل BX، وتشحن داخل السجل ES محتوى الموقعين 789Chو 789Dh.

- غير مباشرة بالسجل. مثال:

LES DI, [BX]

يشحن المعالج السجل DI بالقيمتين [BX] و [BX+1]، كما يشحن السجل ES بالقيمتين [BX+2] و [BX+3].

4-1 تعليمات نقل الرايات

- رمز التعليمة: LAHF

وصف التعليمة:

تنقل التعليمةُ الثمانيّة الدنيا لسجل الرايات في المعالج 8086 وتخزّنها في السجل AH.

أنماط العنونة: بلا حدود

- رمز التعليمة: SAHF

وصف التعليمة:

ينقل المعالج محتوى السجل AH إلى الثمانيّة الدنيا لسجل الرايات.

أنماط العنونة: بلا حدود

- رمز التعليمة: PUSHF

وصف التعليمة:

تدفع هذه التعليمة سجل رايات المعالج (وهو سجل ذو 16 خانة) إلى المكدس، وتنقص مؤشر المكدس بمقدار موقعين.

أنماط العنونة:

- بواسطة السجل.

• رمز التعليمة: POPF

وصف التعليمة:

يستعيد المعالج سجل الرايات من المكدس، ويزيد مؤشر المكدس بمقدار موقعين.

أنماط العنونة:

- بواسطة السجل.

2 التعليمات الحسابية

ويمكن تصنيفها في أربع فئات فرعية:

1-2 تعليمات الجمع

• رمز التعليمة: ADD dst, src

وصف التعليمة:

تجمع التعليمة عدداً محدداً بالمصدر إلى العدد الموجود في الوجهة، وتخزِّن النتيجة في الوجهة. يجب أن يكون المصدر والوجهة من النوع ذاته، أي إما أن يكونا مرمزين على 8 خانات، أو على 16 خانة.

أنماط العنونة:

- بواسطة السجل. مثال:

```
ADD CL, BL      ; instruction operating on 8 bits
ADD CX, BX      ; instruction operating on 16 bits
```

- فورية. مثال:

```
ADD AL, 47h
```

- مباشرة. مثال:

ADD AL, [473Ah]

- غير مباشرة بالسجل. مثال:

ADD AH, Prices[BX]

الانزياح هو ناتج جمع محتوى BX و القيمة Price

- بواسطة الدليل. مثال:

ADD DX, [SI]

يجمع المعالج محتوى السجل DX إلى قيمة ذات انزياح يساوي محتوى السجل SI.

- رمز التعليمة: ADC dst, src

وصف التعليمة:

جمع القيمة المصدر إلى القيمة الوجهة مع الحمل carry، ثم تخزين النتيجة في الوجهة.

أنماط العنونة:

- بواسطة السجل. مثال:

ADC CH, BL

- فورية. مثال:

ADC AX, 1234h

- مباشرة. مثال:

ADC BL, [243Ah]

- غير مباشرة بالسجل. مثال:

ADC AL, Prices[BX]

- بواسطة الدليل. مثال:

ADC DX, [SI]

- رمز التعليمة: INC dst

وصف التعليمة:

زيادة محتوى سجل أو موقع ذاكرة بمقدار 1.

أنماط العنونة:

- بواسطة السجل. مثال:

```
INC BL      ; instruction operating on 8 bits
INC BX      ; instruction operating on 16 bits
```

- غير مباشرة بالسجل. مثال:

INC BYTE PTR[BX]

تجمع واحداً إلى محتوى الذاكرة ذي الانزياح PTR+[BX].

INC WORD PTR[BX]

تجمع واحداً إلى الكلمة المخزّنة في الموقعين ذوي الانزياح [BX] و [BX+1].

INC PRICES[BX]

زيادة محتوى موقع في الجدول Prices ذي الانزياح BX.

- عنونة مباشرة. مثال:

INC MAX_TEMP

زيادة محتوى الموقع MAX_TEMP بمقدار واحد. فإذا كان المتحول MAX_TEMP قد عُرّف كثمانيّة، فإن التعليمة INC تعامله كثمانيّة، وإذا كان ذلك المتحول قد عُرّف ككلمة ذات 16 خانة، فالتعليمة تعامله ككلمة.

- رمز التعليمة: AAA

وصف التعليمة:

ضبط نتيجة جمع رقمين مرمزّين وفق الترميز ASCII.

أنماط العنونة: بلا حدود

مثال

ليكن محتوى السجلين AL و BL قبل تنفيذ التعليمة ADD AL, BL كما يلي:

AL = ASCII 5 = 0011 0101
BL = ASCII 9 = 0011 1001

تصبح النتيجة بعد الجمع: AL=6Eh، وهي نتيجة خاطئة. ولكن بتنفيذ التعليمة AAA يصبح المحتوى:

AL = 000 0100

وخانة الحمل Carry تساوي الواحد. فأصبحت إذن النتيجة صحيحة ومساوية للقيمة 14 وفق الترميز ASCII.

- رمز التعليمة: DAA

وصف التعليمة:

تستخدم هذه التعليمة لضبط جمع عددين مرمزين وفق الترميز BCD. فتفحص هذه التعليمة الأوزان الدنيا Nibble في السجل AL، فإذا كانت أكبر من 9 أو كانت الراية AF مساوية للواحد، يضاف العدد 6 إليها. ثم تفحص التعليمة الأوزان العليا، فإذا كانت أكبر من 9 أو كان هنالك حمل من عملية الجمع السابقة، يضاف العدد 6 إليها، وإلا فتُترك كما هي.

أنماط العنونة: بلا حدود (بواسطة السجل AL حصراً)

مثال:

بفرض أن محتوى السجلين AL و BL قبل تنفيذ التعليمة ADD AL, BL كما يلي:

$AL = 59_{BCD} = 0101\ 1001$

$BL = 35_{BCD} = 0011\ 0101$

فبعد تنفيذ الجمع، يصبح المحتوى:

AL = 8Eh

ولما كانت الأوزان الدنيا تساوي E ، وهي أكبر من القيمة 9، يضاف العدد 6 إليها، فينتج:

$AL = 1001\ 0100 = 94_{BCD}$

وهي نتيجة الجمع الصحيحة وفق الترميز BCD.

2-2 تعليمات الطرح

- رمز التعليمة: SUB dst, src

وصف التعليمة:

طرح العدد المحدد بالمصدر من العدد المحدد بالوجهة، وتخزين النتيجة في الوجهة.

أنماط العنونة:

– فورية. مثال:

SUB AX, 3427h

– بواسطة السجل. مثال:

SUB CX, BX

– غير مباشرة بالسجل. مثال:

SUB CX, table[Bx]

- رمز التعليمة: SBB dst, src

وصف التعليمة:

طرح العدد المحدد بالمصدر من العدد المحدد بالوجهة، وطرح الحمل من النتيجة، ثم التخزين في الوجهة.

أنماط العنونة:

– فورية. مثال:

SBB AX, 3427h

– بواسطة السجل. مثال:

SBB CX, BX

– غير مباشرة بالسجل. مثال:

SBB CX, table[Bx]

- رمز التعليمة: DEC register / memory

وصف التعليمة:

انقاص الوجهة بمقدار 1.

أنماط العنونة:

- بواسطة السجل. مثال:

```
DEC CL      ; instruction operating on 8 bits
DEC BP      ; instruction operating on 16 bits
```

- غير مباشرة بالسجل. مثال:

DEC BYTE PTR [BX]

إنقاص الموقع المحدد بالانزياح PTR + [BX] بمقدار 1 (وهو موقع ذو 8 خانات).

DEC word PTR [BX]

إنقاص بمقدار 1 للكلمة المرمزة على 16 خانة والمخزّنة في الموقعين PTR + [BX] و PTR+1 + [BX].

- رمز التعليمة: NEG dst

وصف التعليمة:

الاستعاضة عن الرقم المحدد بالوجهة بمتممه Complement.

أنماط العنونة:

- بواسطة السجل. مثال:

NEG AL

- غير مباشرة بالسجل. مثال:

NEG WORD Ptr[BP]

- رمز التعليمة: CMP dst, src

وصف التعليمة:

يقارن المعالج ثمانيّة أو كلمة من المصدر بالثمانيّة أو الكلمة المحددة بالوجهة ويرفع الرايات الموافقة. وتختلف هذه التعليمة عن تعليمة الطرح بأنها لاتخزن النتيجة في الوجهة.

أنماط العنونة:

- فورية. مثال:

CMP AL, 01h

- بواسطة السجل. مثال:

CMP BH, CL

- غير مباشرة بالسجل. مثال:

CMP Prices[BX], 49h

- رمز التعليمة: AAS

وصف التعليمة:

تستخدم هذه التعليمة لضبط عملية طرح رقمين مرمزين وفق الترميز ASCII.

أنماط العنونة: بلا حدود

مثال 1:

بفرض أن محتوى السجلين قبل تنفيذ تعليمة الطرح SUB AL, BL هو:

AL = ASCII 9 = 0011 1001
BL = ASCII 5 = 0011 0101

فبعد التنفيذ نجد ما يلي:

AL = 0000 0100 ; 4

CF = 0

وهي نتيجة صحيحة، لذا فإن تنفيذ التعليمة AAS لن يغير في النتيجة.

مثال 2:

بفرض أن محتوى السجلين قبل تنفيذ تعليمة الطرح SUB AL, BL هو:

AL = ASCII 5 = 0011 0101
BL = ASCII 9 = 0011 1001

فبعد التنفيذ نجد ما يلي:

AL = 1111 1100 ; -4

CF = 1

وبتنفيذ التعليمة AAS نجد:

AL = 00000100

CF = 1

فالنتيجة إذن أصبحت صحيحة.

- رمز التعليمة: DAS

وصف التعليمة:

ضبط حاصل طرح رقمين مرمزين وفق الترميز BCD (وعملها مشابه لتعليمة DAA).

أنماط العنونة: بلا حدود (بواسطة السجل AL حصراً)

مثال:

بفرض أن محتوى السجلين قبل تنفيذ تعليمة الطرح SUB AL, BH هو:

$AL = 49_{BCD}$

$BH = 72_{BCD}$

فبعد التنفيذ نجد ما يلي:

AL = D7h

CF = 1

وبتنفيذ التعليمة DAS نجد:

$AL = 77_{BCD}$

CF = 1

ولقد حصلنا على هذه النتيجة بطرح القيمة 6 من الأوزان العليا للسجل AL.

3-2 تعليمات الضرب

- رمز التعليمة: MUL src

وصف التعليمة:

حساب جداء ثمانيّة (أو كلمة) غير مقيدة بإشارة، ومحددة بالمصدر بثمانيّة (أو كلمة) غير مقيدة بإشارة من السجل AX، وتخزين النتيجة في السجل AX (أو في السجلين AX و DX لتخزين الأوزان الدنيا و الأوزان العليا على الترتيب).

أنماط العنونة:

- بواسطة السجل. مثال:

```
MUL BH
MUL CX
```

- غير مباشرة بالسجل. مثال:

```
MUL BYTE Ptr[BX]
```

- رمز التعليمة: IMUL

وصف التعليمة:

جداء ثمانيّة (أو كلمة) ذات إشارة بثمانيّة (أو كلمة) ذات إشارة من السجل AX، وتخزين النتيجة في الوجهة.

أنماط العنونة:

- بواسطة السجل. مثال:

```
IMUL BH
IMUL AX
```

- رمز التعليمة: AAM

وصف التعليمة:

ضبط جداء عددين مرمزين وفق الترميز ASCII، ليكون الناتج متوافقاً مع ذلك الترميز.

أنماط العنونة: بلا حدود

مثال:

بفرض أن محتوى السجلين قبل تعليمة الضرب MUL BH هو:

AL = BCD 5
BL = BCD 9

فبعد التنفيذ نجد ما يلي:

AX = 002Dh

وبتنفيذ التعليمة AAM نجد:

$AX = 0000\ 0100\ 0000\ 0101 = 45_{BCD}$

فالنتيجة إذن أصبحت صحيحة.

4-2 تعليمات القسمة

- رمز التعليمة: DIV src

وصف التعليمة:

تُستخدم هذه التعليمة لتقسيم كلمة (16 خانة) غير مقيدة بإشارة على ثمانيّة، أو لتقسيم كلمة مزدوجة (32 خانة) غير مقيدة بإشارة على كلمة (16 خانة).

عند قسمة كلمة على ثمانيّة، ينبغي أن تكون الكلمة مخزّنة في السجل AX. بعد عملية القسمة، يحتوي السجل AL على القسم الصحيح Quotient، أما باقي القسمة فيُخزن في السجل AH.

أما عند قسمة كلمة مزدوجة، فينبغي أن تكون الأوزان العليا للكلمة في السجل DX، والأوزان الدنيا في السجل AX. بعد القسمة، يحتوي السجل AX على القسم الصحيح، على حين يحتوي السجل DX على باقي القسمة. وفي كلتا الحالتين، يمكن للمقسوم عليه أن يكون موقعاً في الذاكرة أو سجلاً ما في المعالج.

أنماط العنونة:

- بواسطة السجل. مثال:

```
DIV BL
DIV CX
```

- غير مباشرة بالسجل. مثال:

```
DIV BYTE Ptr[BX]
```

- رمز التعليمة: IDIV src

وصف التعليمة:

تؤدي هذه التعليمة إلى قسمة كلمة ذات إشارة على ثمانيّة ذات إشارة، أو قسمة كلمة مزدوجة ذات إشارة على كلمة ذات إشارة. وينطبق عليها ماينطبق على تعليمة DIV من حيث مكان تخزين القاسم والمقسوم عليه.

أنماط العنونة:

- بواسطة السجل. مثال:

```
IDIV BL
```

- غير مباشرة بالسجل. مثال:

```
IDIV Byte Ptr[BX]
```

- رمز التعليمة: AAD

وصف التعليمة:

تهدف هذه التعليمة إلى تحويل رقم ذي 16خانة (ومخزّن في السجل AX) من الترميز BCD إلى الترميز الاثناني المكافئ. ويجب أن تطلب هذه التعليمة قبل تعليمة القسمة.

مثال:

بفرض أن محتوى السجل AX قبل تنقيذ التعليمة AAD هو:

AX = 06 07h (67_{BCD})

فبعد التنفيذ يصبح هذا السجل:

AX = 0043 = 43h (67_{BCD})

والآن يمكن إجراء القسمة على عدد اثناني مخزّن مثلاً في السجل CH. فإذا كانت قيمة هذا السجل CH=09h، نجد بعد تنفيذ التعليمة DIV CH ما يلي:

القسم الصحيح : AL=07

باقي القسمة : AH=04

وهي نتيجة صحيحة.

- رمز التعليمة: CBW

وصف التعليمة:

تحويل ثمانيّة ذات إشارة إلى كلمة ذات إشارة ومرمزة على 16 خانة.

أنماط العنونة:

– بواسطة السجل AX حصراً.

مثال:

بفرض أن محتوى السجل AX قبل التعليمة CBW هو:

AX = 0000 0000 1001 1011 = $-155_{decimal}$

فبعد التنفيذ يصبح هذا السجل:

AX = 1111 1111 1001 1011

- رمز التعليمة: CWD

وصف التعليمة:

تحويل كلمة ذات إشارة إلى كلمة مزدوجة ذات إشارة.

أنماط العنونة:

– تستعمل هذه التعليمة السجلين AX و DX حصراً

مثال:

بفرض أن محتوى السجلين AX و DX قبل التعليمة CWD

هو:

AX=11110000 11000111=-3897 decimal

DX = 0

فبعد التنفيذ يصبح هذان السجلان كما يلي:

AX = 11110000 1100111 = -3897 decimal

DX = 1111 1111 1111 1111

3 التعليمات المنطقية

وتُقسم بدورها إلى ثلاث مجموعات، نعرضها تباعاً.

1-3 التعليمات المنطقية الأساسية

- رمز التعليمة: NOT

وصف التعليمة:

تجري هذه التعليمة عملية عكس منطقية NOT على القيمة المحددة بالوجهة وتخزّنها مكانها.

أنماط العنونة:

– بواسطة السجل. مثال:

NOT BX

– غير مباشرة بالسجل. مثال:

NOT BYTE Ptr [BX]

- رمز التعليمة: AND dst, src

وصف التعليمة:

تجري هذه التعليمة العملية المنطقية AND على كل خانة من الثمانيّة، أو الكلمة المحددة بالمصدر، مع الخانة المقابلة لها من الوجهة، ثم تخزن النتيجة في الوجهة.

أنماط العنونة:

- فورية. مثال:
AND BX, 0DDFh

- بواسطة السجل. مثال:
AND BH, CL

- بواسطة الدليل. مثال:
AND CX, [SI]

- غير مباشرة بالسجل. مثال:
AND AL, BYTE Ptr [BX]

- رمز التعليمة: OR dst, src

وصف التعليمة:

تجري هذه التعليمة العملية المنطقية OR بين خانات الثمانيّة، أو الكلمة المحددة بالمصدر، وتلك المحددة بالوجهة، وتخزن النتيجة في الوجهة.

أنماط العذونة:

- فورية. مثال:
OR BL, 80

- بواسطة السجل. مثال:
OR AH, CL

- غير مباشرة بالسجل. مثال:
OR AH, table[BX]

- بواسطة الدليل. مثال:
OR AH, table[BX] [SI]

- رمز التعليمة: XOR

وصف التعليمة:

تجري هذه التعليمة العملية المنطقية XOR بين خانات الثمانيّة، أو الكلمة المحددة بالمصدر، وتلك المحددة بالوجهة، وتخزن النتيجة في الوجهة.

أنماط العنونة:

- فورية. مثال:

XOR CL, 0Fh

- بواسطة السجل. مثال:

XOR CL, BH

- غير مباشرة بالسجل. مثال:

XOR WORD PTR [BX], AX

• رمز التعليمة: TEST dst, src

وصف التعليمة:

تجري هذه التعليمة العملية المنطقية AND بين المصدر والوجهة، ولكن دون أن تخزّن في الوجهة. أي إنها لاتغير قيمة الوجهة، بل يترافق ذلك مع تغيير رايات المعالج فقط.

أنماط العنونة:

- بواسطة السجل. مثال:

TEST AL, BH

- فورية. مثال:

TEST CX, 0001h

- بواسطة الدليل. مثال:

TEST BP, [BX] [DI]

2-3 تعليمات الإزاحة

• رمز التعليمة: SAL / SHL dst, Count

وصف التعليمة:

إزاحة الثمانيّة (أو الكلمة) المحددة بالوجهة نحو اليسار بالمقدار المحدد بـ Count، وملء الخانات الفارغة بالصفر. (والتعليمتان SHL و SAL متطابقتان، فهما رمزان لتعليمة واحدة).

أنماط العنونة:

- فورية. مثال:

SAL BX, 1

- بواسطة السجل. مثال:

SAL BP, CL

- غير مباشرة بالسجل. مثال:

SAL Byte PTR [BX], 1

• رمز التعليمة: SHR dst, Count

وصف التعليمة:

إزاحة خانات الثمانيّة، أو الكلمة، المحددة بالوجهة نحو اليمين بالمقدار المحدد بـ Count، وملء الخانات الفارغة بالصفر.

أنماط العنونة:

- فورية. مثال:

SHR BP, 1

- بواسطة السجل. مثال:

SHR AL, CL

- غير مباشرة بالسجل. مثال:

SHR BYTE Ptr [BX]

• رمز التعليمة: SAR dst, Count

وصف التعليمة:

إزاحة كل خانة من خانات الكلمة أو الثمانيّة المحددة بالوجهة نحو اليمين بالمقدار Count، مع ملء الخانات الفارغة بقيمها القديمة.

أنماط العنونة:

- فورية. مثال:

SAR DI, 1

إن قيمة الوزن الأعلى MSB الجديدة مساوية لقيمتها القديمة.

- غير مباشرة بالسجل. مثال:

SAR Word Ptr [BP], CL

3-3 تعليمات الدوران

• رمز التعليمة: ROL dst, Count

وصف التعليمة:

تدوير الثمانية (أو الكلمة) المحددة بالوجهة يساراً عدداً من الخانات قدره Count.

أنماط العنونة:

- فورية. مثال:

ROL AX, 1

- بواسطة السجل. مثال:

ROL BL, CL

- غير مباشرة بالسجل. مثال:

ROL FACTOR[BX], 1

• رمز التعليمة: ROR dst, Count

وصف التعليمة:

تدوير الثمانية أو الكلمة المحددة بالوجهة يميناً عدداً من الخانات قدره Count. إن الخانة الدنيا تكتب في الراية CF وفي الخانة العليا MSB.

أنماط العنونة:

- فورية. مثال:

ROR BL

- بواسطة السجل:

ROR AL, CL

- غير مباشرة بالسجل:

ROR WORD PTR [BX], CL

- رمز التعليمة: RCL dst, Count

وصف التعليمة:

تدوير الثمانيّة (أو الكلمة) المحددة بالوجهة يساراً عدداً من الخانات قدره Count. نلاحظ أن الفرق هنا هو أن الخانة العليا تمر، عند تدويرها، عبر راية الحمل، وتؤخذ قيمة راية الحمل إلى الخانة الدنيا. وبكلمة أخرى، فالتدوير يتم عبر الحمل.

أنماط العنونة:

- فورية. مثال:

RCL DX, 1

- بواسطة السجل. مثال:

RCL AX, CL

- غير مباشرة بالسجل. مثال:

RCL SUM[BX], CL

- رمز التعليمة: RCR dst, src

وصف التعليمة:

تدوير الثمانيّة (أو الكلمة) المحددة بالوجهة يميناً عدداً من الخانات قدره Count. نلاحظ أن الفرق هنا هو أن الخانة الدنيا تمر، عند تدويرها، عبر راية الحمل، وتؤخذ قيمة راية الحمل إلى الخانة العليا. وبكلمة أخرى، فيجري التدوير بالحمل.

أنماط العنونة:

- فورية. مثال:

RCR BX, 1

- بواسطة السجل. مثال:

RCR AX, CL

- غير مباشرة بالسجل. مثال:

RCR BYTE Ptr[BX], CL

4 تعليمات سلاسل المحارف

نسمي سلسلة محارف مجموعة من الثمانيّات والكلمات المخزّنة في مواقع متتابعة من الذاكرة، وهي تحتوي غالباً على رموز ASCII.

- رمز التعليمة: REPNZ / REPNE / REPZ / REPE / REP

وصف التعليمة:

إن REP هو سابقة Prefix تسبق تعليمات السلاسل، وهي تشحن السجل CX بطول السلسلة، وتكرر التعليمة مع إنقاص السجل CX بمقدار 1 في كل مرة، إلى أن يصبح محتوى السجلCX معدوماً. فمثلاً تؤدي العبارة REP MOVSB إلى تنفيذ تعليمة نقل لثمانيّات سلسلة المحارف، تنفيذاً متكرراً إلى أن يصبح محتوى السجل CX (الذي شُحن بطول السلسلة) معدوماً.

أما السوابق التالية: REPNZ / REPNE / REPZ / REPE / REP فهي تكرر تنفيذ التعليمة إلى أن يتحقق أحد شرطين:

الشرط الأول: أن يصبح محتوى السجل CX صفراً.

الشرط الثاني: ويختلف باختلاف السابقة:

- REPE (REPeat if Equal): يتكرر التنفيذ مادامت القيمة المقروءة مساوية لتلك المخزّنة في السجل AX.
- (REPeat if Not Equal) REPNE: يتكرر التنفيذ مادامت القيمة المقروءة لا تساوي تلك المخزّنة في السجل AX.
- (REPeat if Zero) REPZ: يتكرر التنفيذ مادامت القيمة المقروءة معدومة.
- (REPeat if Not Zero) REPNZ: يتكرر التنفيذ مادامت القيمة المقروءة غير معدومة.

مثال:

```
REPNE SCASW
```

تمسح هذه التعليمة سلسلة من الكلمات إلى أن تصبح الكلمة المقروءة مساوية للكلمة المخزنة في السجل AX، أو إلى أن تنتهي سلسلة الكلمات.

- رمز التعليمة: MOVS / MOVSB / MOVSW

وصف التعليمة:

تنسخ هذه التعليمة ثمانيّة أو كلمة من موقع في قطاع المعطيات إلى موقع في القطاع الإضافي. يعطى انزياح المصدر بالمؤشر SI وانزياح الوجهة بالسجل DI. وفي حال التكرار، يُخزّن عدد المرات في السجل CX. وتتغير قيمة المؤشرين SI و DI آلياً بعد كل عملية نقل، فتزداد قيمتهما إذا كان محتوى الراية DF مساوياً الصفر، وتنقص قيمتهما في الحالة المعاكسة. ولما كان بالإمكان نقل ثمانيّات أو كلمات، فمن الواجب توجيه المجمِّع لأداء العملية المطلوبة. ويحدث ذلك باختيار التعليمة المناسبة، فمثلاً تدل التعليمة MOVSB على ضرورة نقل ثمانيّات، في حين تنقل التعليمة MOVSW كلمات مرمّزة على 16 خانة.

مثال:

```
CLD               ; clear DF
MOV AX, 00h
MOV DS, AX        ; initialize data segment register to 0
MOV ES, AX        ; initialize extra segment register to 0
MOV SI, 2000h     ; load offset of start of source string into SI
MOV DI, 2400h     ; load offset of start of destination string into DI
MOV CX, 04h       ; load length of string in CX as counter
REP MOVSB         ; decrement CX and MOVSB until CX=0
```

- رمز التعليمة CMPS / CMPSB / CMPSW

وصف التعليمة:

تقارن هذه التعليمة ثمانيّة أو كلمة من السلسلة المصدر بثمانيّة أو كلمة من السلسلة الوجهة. تستخدم هذه التعليمة السجل

SI ليؤشر على المصدر، و السجل DI ليؤشر على الوجهة. ويعبر عن نتيجة المقارنة بسجل الرايات. تتغير قيمة السجلين SIو DI آلياً تبعاً لراية الاتجاه DF. ويُفترض عند التنفيذ أن الوجهة تنتمي إلى القطاع الإضافي وأن المصدر موجود في قطاع المعطيات.

مثال:

```
MOV CX, 100         ; put number of string elements in CX
MOV SI, 2000h       ; load offset of start of source string into SI
MOV DI, 2400h       ; load offset of start of destination string into DI
STD                 ; DF=1 so SI and DI will auto decrement
                    ; after compare
REPE CMPSB          ; repeat comparaison of string bytes until end
                    ; or compared bytes are not equal
```

- رمز التعليمة: INS / INSB / INSW

وصف التعليمة:

تنقل هذه التعليمة ثمانيّة أو كلمة من معبر محدد بالمصدر إلى موقع في القطاع الإضافي، يؤشر عليه السجل DI. يُحدد عنوان المعبر المصدر بالسجل DX، ويتغير محتوى السجل DI آلياً تبعاً لراية الاتجاه، فتزداد قيمته إذا كان محتوى الراية معدوماً، وينقص في الحالة المعاكسة. تُستخدم التعليمة INSB لنقل ثمانيات إلى الذاكرة، في حين تنقل التعليمة INSW كلمات مرمزة على 16خانة.

مثال:

```
CLD                     ; clear DF to autoincrement DI
MOV DI, OFFSET BUF      ; point DI to input buffer
MOV DX, 0FFF8h          ; load DX with port address
MOV CX, LENGTH BUF      ; load number of bytes to be read in CX
REP INSB DX             ; copy bytes from port until buffer full
```

- رمز التعليمة: OUTS / OUTSB / OUTSW

وصف التعليمة:

تنقل هذه التعليمة ثمانيّة أو كلمة من موقع في القطاع الإضافي، يؤشر عليه السجل SIإلى معبر محدد بالسجل DX، ويتغير

محتوى السجل SI آلياً تبعاً لراية الاتجاه، فتزداد قيمته إذا كان محتوى الراية معدوماً، وينقص في الحالة المعاكسة. تُستخدم التعليمة OUTSB لنقل ثمانيّات إلى الذاكرة، في حين تنقل التعليمة OUTSW كلمات مرمّزة على 16خانة.

مثال:

```
CLD                     ; clear DF to autoincrement DI
MOV DI, OFFSET BUF      ; point DI to input buffer
MOV DX, 0FFF8h          ; load DX with port address
MOV CX, 100             ; load number of bytes to be output in CX
REP OUTSB DX            ; copy bytes to port until buffer empty
```

- رمز التعليمة: SCAS / SCASB / SCASW

وصف التعليمة:

تقارن هذه التعليمة ثمانيّة أو كلمة مخزّنة في السجل AL أو السجل AX بثمانيّة أو كلمة من السلسلة الوجهة. تستخدم هذه التعليمة السجل DI ليؤشر على الوجهة التي ينبغي أن تكون في القطاع الإضافي. ويُعبر عن نتيجة المقارنة بسجل الرايات. تتغير قيمة السجل DI آلياً تبعاً لراية الاتجاه DF، فإذا كانت معدومة ازدادت القيمة بعد كل تعليمة، وإلا فإنها تُنقص.

مثال:

نود البحث عن الثمانية ODh في سلسلة ذات 80 محرفاً.

```
MOV AL, 0Dh                  ; byte to be scanned for into AL
MOV DI, OFFSET TEXT_STRING   ; offset of string to DI
MOV CX, 80                   ; CX used as element counter
CLD                          ; clear DF so DI autoincrement
REPNE SCAS TEXT_STRING       ; compare byte in string
                             ; with byte in AL
```

- رمز التعليمة: LODS / LODSB / LODSW

وصف التعليمة:

تنقل هذه التعليمة ثمانيّة أو كلمة من موقع يؤشر عليه

السجل SI إلى السجل AL أو AX. تتغير قيمة السجل SI آلياً تبعاً لراية الاتجاه DF. فإذا كانت معدومة ازدادت القيمة بعد التنفيذ، وإلا فإنها تنقص. تُستخدم التعليمة LODSB لنقل الثمانيّات، في حين تؤدي التعليمة LODSW إلى نقل كلمات مرمّزة على 16خانة.

مثال:

```
CLD                                ; clear DF so SI
                                   ; will autoincrement
MOV SI, OFFSET SOURCE_STRING       ; point SI at string
LODS SOURCE_STRING
```

- رمز التعليمة: STOS / STOSB / STOSW

وصف التعليمة:

تنقل هذه التعليمة ثمانيّة أو كلمة من السجل AL أو AX إلى موقع في القطاع الإضافي يؤشر عليه السجل DI. تتغير قيمة السجل DI آلياً تبعاً لراية الاتجاه DF. فإذا كانت معدومة ازدادت القيمة بعد التنفيذ، وإلا فإنها تنقص. تُستخدم التعليمة STOSB لنقل الثمانيّات، في تؤدي أن التعليمة STOSW إلى نقل كلمات مرمّزة على 16خانة.

مثال:

```
CLD                                ; clear DF so SI
                                   ; will autoincrement
MOV DI, OFFSET TARGET_STRING       ; point DI
                                   ; at destination string
STOS TARGET_STRING
```

5 تعليمات التحكم في تسلسل التنفيذ

1-5 تعليمات الاستدعاء والعودة

• رمز التعليمة: Call

وصف التعليمة:

تُستخدم هذه التعليمة لاستدعاء إجرائية من النمط القريب أو البعيد. عندما ينفذ المعالج تعليمة Call من النمط القريب فإنه ينقص مؤشر المكدس بمقدار 2، ويشحن عنوان التعليمة التالية في المكدس، ويشحن سجل التعليمات IP بانزياح أول تعليمة في الإجرائية. أما تعليمة Call من النمط البعيد فهي تنقص مؤشر المكدس بمقدار 2، وتشحن عنوان السجل CS في المكدس، ثم تنقص المؤشر بمقدار 2 ثانيةً، وتشحن في المكدس انزياح التعليمة التالية. وأخيراً، تشحن السجل CS بعنوان أول تعليمة في الإجرائية.

مثال:

```
CALL BX                 ; BX contains the offset of the first
                        ; instruction in the procedure
CALL WORD PTR [BX]      ; offset is in 2 memory locations in DS
CALL SMART_DIVIDE       ; SMART_DIVIDE is the name
                        ; of the procedure
```

• رمز التعليمة: RET

وصف التعليمة:

تعيد هذه التعليمة المعالج من الإجرائية لينفذ التعليمة التي تلي التعليمة Call. عند تنفيذ هذه التعليمة، يسترجع المعالج قيمة مؤشر التعليمات IP من المكدس. وفي حال استدعاء إجرائية من النمط البعيد، يسترجع المعالج أيضاً السجل CS من المكدس.

ومن الممكن إلحاق قيمة في التعليمة RET، مثلاً RET 6، وهذا ما يؤدي إلى زيادة مؤشر المكدس ستة مواقع إضافية.

2-5 تعليمات القفز

• رمز التعليمة: JMP

وصف التعليمة:

تدفع هذه التعليمة المعالج للقفز إلى العنوان المحدد بالتعليمة. فإذا كان القفز إلى عنوان في القطاع ذاته، وجب شحن مؤشر التعليمات IP فقط لتحقيق القفز. أما إذا كان القفز إلى عنوان في قطاع آخر، فيجب عندئذٍ تغيير مؤشر التعليمات IP وسجل قطاع البرنامج CS معاً.

مثال:

```
JMP CONTINUE           ; jump to line whose label is CONTINUE
JMP BX                 ; replace the contents of IP with BX
JMP WORD PTR [BX]      ; replace IP with a word from memory
```

• رمز التعليمة: JA / JNBE

Jump if Above / Jump if Not Below nor Equal

وصف التعليمة:

لهذه التعليمة اسمان: JA و JNBE. وتُستخدم بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الصفر ZF وراية الحمل CF تساويان الصفر، قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت الرايتان مختلفتين عن الصفر تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
CMP AX, 4371h     ; compare by substracting 4371h from AX
JA RUN_PRESS      ; jump to label RUN_PRESS If AX above 4371h
```

• رمز التعليمة: JAE / JNB / JNC

Jump if Above or Equal / Jump if Not Below / Jump if No Carry

وصف التعليمة:

لهذه التعليمة ثلاثة أسماء: JAE و JNB و JNC. وتُستخدم بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الحمل CF تساوي الصفر، قفز المعالج إلى العنوان المحدّد بالتعليمة. و إذا كانت الراية مختلفة عن الصفر تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
CMP AX, 4371h        ; compare by substracting 4371h from AX
JAE RUN_PRESS        ; jump to label RUN_PRESS
                     ; If AX above or equal 4371h
```

• رمز التعليمة: JB / JC / JNAE

Jump if Below / Jump if Carry / Jump if Not Above nor Equal

وصف التعليمة:

لهذه التعليمة ثلاثة أسماء: JNAE و JB و JC. وتُستخدم بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الحمل CF تساوي الواحد قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت الراية مساوية الصفر تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
CMP AX, 4371h     ; compare by substracting 4371h from AX
JB RUN_PRESS      ; jump to label RUN_PRESS If AX below 4371h
```

• رمز التعليمة: JBE / JNA

Jump if Below or Equal / Jump if Not Above

وصف التعليمة:

لهذه التعليمة اسمان: JNA و JBE. تُستخدم هذه التعليمة

بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الحمل CF أو راية الصفر ZF تساوي الواحد قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت الرايتان تساويان الصفر تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
CMP AX, 4371h        ; compare by substracting 4371h from AX
JBE RUN_PRESS        ; jump to label RUN_PRESS
                     ; If AX not above 4371h
```

- رمز التعليمة: JE / JZ

Jump if Equal / Jump if Zero

وصف التعليمة:

لهذه التعليمة اسمان: JE و JZ. وتُستخدم بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الصفر ZF تساوي الواحد قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت الراية تساوي الصفر تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
AGAIN:
    CMP BX, DX          ; compare by substracting DX from BX
    JE DONE             ; jump If BX=DX to label DONE
    SUB BX, AX          ; Else substract AX
    INC CX              ; increment counter
    JMP AGAIN           ; check again
DONE:
    MOV AX, CX          ; copy count to AX
    IN AL, 8Fh          ; read data from port 8Fh
    SUB AL, 30h         ; substract minimum value
    JZ START_MACHINE    ; jump to label If result of substraction is 0
```

- رمز التعليمة: JG / JNLE

Jump if Greater / Jump if Not Less than nor Equal

وصف التعليمة:

لهذه التعليمة اسمان: JG و JNLE. وتُستخدم بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الصفر مساوية للصفر وراية الحمل CF مساوية لراية الفائض OV قفز المعالج إلى العنوان المحدّد بالتعليمة. وإلا، تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين -128 و 127 ثمانيّة.

مثال:

```
CMP BL, 39h          ; compare by substracting 39h from BL
JG SHORT_LABEL       ; jump to label If BL more positive than 39h
```

- رمز التعليمة: JGE / JNL

Jump if Greater or Equal / Jump if Not Less

وصف التعليمة:

لهذه التعليمة اسمان: JGE و JNL. وتُستخدم بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الإشارة SF مساوية لراية الفائض OV قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت الرايتان مختلفتين تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين -128 و 127 ثمانيّة.

مثال:

```
CMP BL, 39h            ; compare by substracting 39h from BL
JGE SHORT_LABEL        ; jump to label
                       ; If BL more positive (or equal) than 39h
```

- رمز التعليمة: JL / JNGE

Jump if Less / Jump if Not Greater nor Equal

وصف التعليمة:

لهذه التعليمة اسمان: JL و JNGE. وتُستخدم بعد تعليمة

المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الإشارة SF تختلف عن راية الفائض OV قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت الرايتان متساويتين تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128-و 127 ثمانيّة.

مثال:

```
CMP BL, 39h          ; compare by substracting 39h from BL
JL SHORT_LABEL       ; jump to label If BL more negative than 39h
```

- رمز التعليمة: JLE / JNG

Jump if Less or Equal / Jump if Not Greater

وصف التعليمة:

لهذه التعليمة اسمان: JLE و JNG. وتُستخدم بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الصفر ZF مساوية الواحد، أو راية الإشارة SF تختلف عن راية الفائض OV قفز المعالج إلى العنوان المحدّد بالتعليمة. وإلا تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
CMP BL, 39h          ; compare by substracting 39h from BL
JLE SHORT_LABEL      ; jump to label
                     ; If BL more negative (or equal) than 39h
```

- رمز التعليمة: JNE / JNZ

Jump if Not Equal / Jump if Not Zero

وصف التعليمة:

لهذه التعليمة اسمان: JNE و JNZ. وتُستخدم بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الصفر ZF مساوية الصفر قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت راية الصفر ZF مساوية الواحد تابع المعالج عمله دون قفز.

ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
AGAIN:
    IN AL, 0F8h       ; read data from port
    CMP AL, 72        ; compare by substracting 72 from AL
    JNE AGAIN         ; jump to label AGAIN If AL not equal 72
```

- رمز التعليمة: JNO

Jump if No Overflow

وصف التعليمة:

تُستخدم هذه التعليمة بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الفائض OF مساوية الصفر قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت تلك الراية مساوية الواحد تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
    ADD AL, BL        ; add signed bytes in AL and BL
    JNO DONE          ; process DONE If no overflow
    MOV AL, 0h        ; Else, load error code in AL
DONE:
    OUT 24h, AL       ; send result to display
```

- رمز التعليمة: JNP / JPO

Jump if No Parity / Jump if Parity Odd

وصف التعليمة:

لهذه التعليمة اسمان: JPO و JNP. نقول عن ثمانيّة أنها فردية إذا كان عدد الأرقام '1' فيها عدداً فردياً. فإذا كانت نتيجة تنفيذ تعليمة ما فردية كُتبت القيمة 0 في راية التثبت في المعالج. تدفع التعليمة JNP المعالج للقفز إلى العنوان المحدّد بها إذا كانت راية التثبت PF مساوية الصفر. وإذا كانت الراية PF مساوية الواحد تابع

المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
IN AL, 0F8h         ; read data from port
OR AL, AL           ; set flags
JPO ERR_MESSAGE     ; send error message If odd parity found
```

- رمز التعليمة: JNS

Jump if Not Signed

وصف التعليمة:

تُستخدم هذه التعليمة بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الإشارة SF مساوية الصفر قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت تلك الراية مساوية الواحد تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
DEC AL          ; decrement counter
JNS REDO        ; jump to label REDO
                ; If counter has not decremented to FFh
```

- رمز التعليمة: JO

Jump if Overflow

وصف التعليمة:

تُستخدم هذه التعليمة بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الفائض OF مساوية الواحد قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت تلك الراية تساوي الصفر تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
ADD AL, BL          ; add signed bytes in AL and BL
JO ERROR            ; jump to Error If overflow from Add
MOV SUM, AL         ; Else, put result in memory location called SUM
```

- رمز التعليمة: JP / JPE

Jump if Parity / Jump if Parity Even

وصف التعليمة:

لهذه التعليمة اسمان: JP و JPE. نقول عن ثمانيّة أنها زوجية إذا كان عدد الأرقام '1' فيها عدداً زوجياً. فإذا كانت نتيجة تنفيذ تعليمة ما فردية كُتبت القيمة 1 في راية التثبت في المعالج. تدفع التعليمة JP المعالج للقفز إلى العنوان المحدّد بها إذا كانت راية التثبت PF مساوية الواحد. وإذا كانت الراية PF تساوي الصفر تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
IN AL, 0F8h         ; read data from port
OR AL, AL           ; set flags
JPE ERR_MESSAGE     ; send error message If even parity found
```

- رمز التعليمة: JS

Jump if Signed

وصف التعليمة:

تُستخدم هذه التعليمة بعد تعليمة المقارنة أو أي تعليمة أخرى تؤثر على الرايات. فإذا كانت راية الإشارة SF مساوية الواحد قفز المعالج إلى العنوان المحدّد بالتعليمة. وإذا كانت تلك الراية تساوي الصفر تابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
ADD BL, DH          ; add signed byte in DH to signed byte in BL
JS TOO_COLD         ; jump to label TOO_COLD
                    ; If result of addition is negative number
```

- رمز التعليمة: JCXZ

Jump if the CX register is Zero

وصف التعليمة:

تؤدي هذه التعليمة إلى القفز إلى العنوان المحدّد بها إذا كان محتوى السجل CX مساوياً الصفر. وإلا، يتابع المعالج عمله دون قفز. ومن الجدير بالذكر أن مسافة القفز يجب أن تكون محصورة بين 128- و 127 ثمانيّة.

مثال:

```
    JCXZ SKIP_PROCESS       ; If CX=0 skip over process
COUNT:
    SUB [BX], 07h           ; substract from data value
    INC BX                  ; point ot next value
    LOOP COUNT              ; loop until CX=0
SKIP_COUNT: ...             ; next instruction
```

3-5 تعليمات التكرار

تُستخدم هذه التعليمات لتكرار تنفيذ مجموعة من التعليمات عدداً من المرات.

- رمز التعليمة: LOOP

وصف التعليمة:

تُستخدم هذه التعليمة لتكرار تنفيذ مجموعة من التعليمات عدداً من المرات. يُخزّن عدد المرات في السجل CX، ويُنقص هذا السجل مع كل تكرار إلى أن تنعدم قيمته. وينبغي على عنوان التكرار أن يبعد مسافة تقع بين 128- و 127 ثمانيّة فقط.

مثال:

```
    MOV BX, OFFSET PRICES      ; point BX at first element in array
    MOV CX, 40                 ; load CX with number of elements
                               ; in array
NEXT:
    MOV AL, [BX]               ; get element from array
    ADD AL, 07                 ; add correction factor
    DAA                        ; decimal adjust result
    MOV [BX], AL               ; put result back in array
    LOOP NEXT                  ; repeat until all elements adjusted
```

- رمز التعليمة: LOOPE / LOOPZ

وصف التعليمة:

لهذه التعليمة اسمان: LOOPZ و LOOPE. وتُستخدم لتكرار تنفيذ مجموعة من التعليمات عدداً من المرات أو إلى أن تصبح راية الصفر ZF مساوية الصفر. يُخزّن عدد المرات في السجل CX، ويُنقص هذا السجل مع كل تكرار. يتوقف التنفيذ عند تحقق أحد الشرطين التاليين:

- التنفيذ إلى أن يصبح محتوى السجل CXمعدوماً؛
- التنفيذ إلى أن تصبح راية الصفر ZF مساوية الصفر.

وينبغي على عنوان التكرار أن يبعد مسافة تقع بين 128- و 127 ثمانيّة فقط.

مثال:

```
    MOV BX, OFFSET ARRAY       ; point BX at start of array
    DEC BX
    MOV CX, 100                ; put number of elements in CX
NEXT:
    INC BX                     ; point to next element in array
    CMP [BX], 0FFh             ; compare array element with FFh
    LOOPE NEXT
```

- رمز التعليمة: LOOPNE / LOOPNZ

وصف التعليمة:

لهذه التعليمة اسمان: LOOPNZ و LOOPNE. وتُستخدم لتكرار تنفيذ مجموعة من التعليمات عدداً من المرات أو إلى أن تصبح راية الصفر ZF مساوية الواحد. يُخزّن عدد المرات في السجل CX، وينُقص هذا السجل مع كل تكرار. يتوقف التنفيذ عند تحقق أحد الشرطين التاليين:

- التنفيذ إلى أن يصبح محتوى السجل CXمعدوماً؛
- التنفيذ إلى أن تصبح راية الصفر ZF مساوية الواحد.

وينبغي على عنوان التكرار أن يبعد مسافة تقع بين 128- و 127 ثمانيّة فقط.

مثال:

```
    MOV BX, OFFSET ARRAY      ; point BX at start of array
    DEC BX
    MOV CX, 100               ; put number of elements in CX
NEXT:
    INC BX                    ; point to next element in array
    CMP [BX], 0Dh             ; compare array element with 0Dh
    LOOPNE NEXT
```

4-5 تعليمات المقاطعة

تسمح هذه التعليمات بمقاطعة عمل المعالج، فيقفز إلى برنامج فرعي معين وينفذه، ثم يعود لمتابعة ماكان يقوم به قبل المقاطعة.

- رمز التعليمة: INT

وصف التعليمة:

يستدعي المعالج عند تنفيذ هذه التعليمة إجرائية بعيدة، يختلف عنوانها بحسب رقم النوع المحدد بالتعليمة. يؤدي المعالج عند تنفيذ هذه التعليمة المهمات التالية:

- ينقص مؤشر المكدس بمقدار 2، ويدفع سجل الرايات إلى المكدس.
- ينقص مؤشر المكدس بمقدار 2، ويدفع سجل قطاع البرنامج CS إلى المكدس.
- ينقص مؤشر المكدس بمقدار 2، ويدفع انزياح عنوان التعليمة اللاحقة إلى المكدس.
- يحصل على قيمة جديدة لمؤشر التعليمات IP من موقع الذاكرة ذي العنوان: 4×(قيمة النوع).
- يحصل على قيمة جديدة سجل قطاع البرنامج CS من موقع الذاكرة ذي العنوان: 4×(قيمة النوع) + 2.
- يمحو كلاً من الرايتين TF وIF.

مثال:

```
INT 35        ; new IP from 0008Ch and new CS from 0008Eh
```

• رمز التعليمة: INTO

وصف التعليمة:

إذا كانت راية الفائض مساوية الواحد، يستدعي المعالج عند تنفيذ هذه التعليمة إجرائية بعيدة لمعالجة الفائض. يؤدي المعالج عند تنفيذ هذه التعليمة المهمات التالية:

- ينقص مؤشر المكدس بمقدار 2، ويدفع سجل الرايات إلى المكدس.
- ينقص مؤشر المكدس بمقدار 2، ويدفع سجل قطاع البرنامج CS إلى المكدس.
- ينقص مؤشر المكدس بمقدار 2، ويدفع انزياح عنوان التعليمة اللاحقة إلى المكدس.
- يحصل على قيمة جديدة لمؤشر التعليمات IP من موقع الذاكرة ذي العنوان 00010h.

- يحصل على قيمة جديدة لسجل قطاع البرنامج CS من موقع الذاكرة ذي العنوان 00012h.
- يمحو كلاً من الرايتين TF وIF.

مثال:

```
INTO            ; call interrupt procedure If OF=1
```

- رمز التعليمة: IRET

وصف التعليمة:

تُستخدم التعليمة IRET في نهاية إجرائية المقاطعة لإعادة السيطرة إلى البرنامج الرئيسي. ولذا، يقوم المعالج بالعمليات التالية:

- استرجاع قيمة سجل قطاع البرنامج CS من المكدس.
- استرجاع قيمة مؤشر التعليمات IP من المكدس.
- استرجاع محتوى سجل الرايات.

مثال:

```
IRET     ; return from Interrupt handler
```

6 تعليمات التحكم في المعالج

تقسم تعليمات التحكم في المعالج Processor Control إلى ثلاث مجموعات فرعية، نعرضها تباعاً.

1-6 تعليمات الرايات

- رمز التعليمة: CLC

وصف التعليمة:

تمحي هذه التعليمة راية الحمل CF.

مثال:

```
CLC      ; clear carry flag
```

- رمز التعليمة: CLD

وصف التعليمة:

تمحي هذه التعليمة راية الاتجاه DF. فإذا أصبحت الراية تساوي الصفر يزداد المؤشران SI و DI تلقائياً بعد تنفيذ إحدى تعليمات المحارف، مثل MOVS, CMPS, SCAS.

مثال:

```
CLD	; clear direction flag
```

- رمز التعليمة: CLI

وصف التعليمة:

تمحي هذه التعليمة راية المقاطعة IF. فإذا كانت الراية تساوي الصفر فلن يستجيب المعالج لأي إشارة مقاطعة ترِد على المدخل INTR.

مثال:

```
CLI	; clear interrupt flag
```

- رمز التعليمة: CMC

وصف التعليمة:

تؤدي هذه التعليمة إلى استبدال القيمة المتمِّمة لراية الحمل CF بالقيمة الحالية. فإذا كانت القيمة صفراً أصبحت واحداً والعكس بالعكس.

مثال:

```
CMC	; invert carry flag
```

- رمز التعليمة: STC

وصف التعليمة:

تؤدي هذه التعليمة إلى كتابة القيمة '1' في راية الحمل CF.

مثال:

```
STC      ; set carry flag
```

- رمز التعليمة: STD

وصف التعليمة:

تكتب هذه التعليمة القيمة '1' في راية الاتجاه DF. فإذا أصبحت الراية تساوي الواحد، يُنقص المؤشران SI و DI تلقائياً بعد تنفيذ إحدى تعليمات الحارف، مثل MOVS, CMPS, SCAS.

مثال:

```
STD      ; set direction flag
```

- رمز التعليمة: STI

وصف التعليمة:

تكتب هذه التعليمة القيمة '1' في راية المقاطعة IF. فإذا كانت الراية تساوي الواحد فإن المعالج سيستجيب لأي إشارة مقاطعة ترد على المدخل INTR.

مثال:

```
STI      ; set interrupt flag
```

2-6 تعليمات التزامن مع الإشارات الخارجية

- رمز التعليمة: ESC

وصف التعليمة:

تُستخدم هذه التعليمة في تمرير تعليمات إلى المعالج الحسابي مثل المعالج 8087. تُرمز تعليمات المعالج الحسابي على ست خانات وتُضمَّن في التعليمة ESC. فعندما يجلب المعالج 8086 التعليمة ESC من الذاكرة، فإن المعالج الحسابي يجلب التعليمة ذاتها ويخزنها في رتله، لأنه يشترك مع المعالج 8086 في المسرى. بعد تحليل التعليمة ESC، ينفذ المعالج الحسابي المهمة المطلوبة منه بحسب

الرمز المضمّن في التعليمة. في حين ينظر المعالج 8086 إلى التعليمة ESC كأنها تعليمة NOP.

- رمز التعليمة: HLT

وصف التعليمة:

تدفع هذه التعليمة المعالج 8086 إلى التوقف عن جلب المزيد من التعليمات. ولايخرج المعالج من هذه الحالة إلا بورود إشارة مقاطعة على المدخل INTR أو المدخل NMI أو إعادة إقلاع المعالج بإشارة Reset خارجية.

- رمز التعليمة: LOCK

وصف التعليمة:

تفيد هذه التعليمة في النظم الحاسوبية المتعددة المعالجات. تشترك هذه المعالجات ببعض الموارد المادية مثل القرص الصلب والذاكرة. وللحيلولة دون حدوث تضارب بين هذه المعالجات، يستطيع كل معالج إقفال المورد المشترك بحيث يمنع بقية المعالجات من الوصول إليه في تلك الفترة. يتحقق إقفال مورد ما بواسطة التعليمة LOCK.

مثال:

```
LOCK XCHG SEMAPHORE, AL ; XCHG needs two bus accesses
     ; using LOCK prevents other microprocessors
     ; from taking control between accesses
```

- رمز التعليمة: WAIT

وصف التعليمة:

عند تنفيذ هذه التعليمة، يدخل المعالج في حالة ركود Idle State إلى أن تظهر إشاره على المدخل TEST، أو ورود مقاطعة INTR أو NMI. تفيد هذه التعليمة في تزامن المعالج مع حوادث خارجية، مثل المعالج الحسابي.

3-6 تعليمة بلا عمل

• رمز التعليمة: NOP

وصف التعليمة:

تستهلك هذه التعليمة ثلاثة أدوار من الساعة الخارجية؛ وهي تعليمة بلا عمل، ولا تؤثر على سجل الرايات. وتفيد هذه التعليمة، مثلاً، في تكوين حلقات التأخير.

مثال:

```
NOP     ; no operation instruction
```



الجلسات العملية

يقسم البرنامج العملي إلى قسمين:

- الجلسات العملية على الدارات المنطقية (5 جلسات)

1 تعرّف برنامج المحاكاة المنطقية

- تحقيق مجموعة دوال منطقية
- إنشاء المخطط
- إجراء المحاكاة
- تحليل النتائج

2 بناء نظام تركيبي:

– جمع عددين وعرض النتائج على وحدات إظهار

- تصميم الدارة
- إنشاء المخطط
- التنفيذ والتحليل

3 بناء نظام تتابعي:

– جمع عددين ثم ضربهما بعدد ثالث

- تصميم الدارة (باستخدام العدادات وسجلات الانزياح)
- إنشاء المخطط
- التنفيذ والتحليل

4–5 مسألة تتابعية متزا...

– تحليل وتنفيذ آلة منتهية الحالات

- تصميم الداره
- إنشاء المخطط
- التنفيذ والتحليل

الجلسات العملية على البرمجة بلغة المجمّع *(6 جلسات)*

1 تعرّف برنامج المجمّع وبرنامج التنقيح:

- كتابة برنامج يقوم بجمع قيمتين وتخزين النتيجة
  - كتابة البرنامج على محرر النصوص
  - تجميع البرنامج
  - تنقيح البرنامج
  - اختبار البرنامج

2 استخدام التعليمات الحسابية والمنطقية:

- كتابة برنامج لحساب الوسطي المثقّل
  - كتابة البرنامج
  - تجميع البرنامج
  - تنقيح البرنامج
  - اختبار البرنامج

3 استخدام الرايات:

- ترتيب أرقام مخزّنة في الذاكرة ترتيباً تصاعدياً
- إيجاد القيمة العظمى لمجموعة أرقام في الذاكرة
  - كتابة البرنامج
  - تجميع البرنامج
  - تنقيح البرنامج
  - اختبار البرنامج

4 استخدام الحلقات والشروط

- حساب الأرقام في سلسلة فيبوناتشي Fibonacci
- فحص رقم معطى لمعرفة ما إذا كان أولياً
  - كتابة البرنامج
  - تجميع البرنامج
  - تنقيح البرنامج
  - اختبار البرنامج

5 استخدام سلاسل المحارف

- ترتيب سلسلة محارف ترتيباً عكسياً
- البحث عن محرف معيّن ضمن سلسلة محارف
  - كتابة البرنامج
  - تجميع البرنامج
  - تنقيح البرنامج
  - اختبار البرنامج

6 استخدام المقاطعات

- إرسال ملف معيّن إلى الطابعة
  - كتابة البرنامج
  - تجميع البرنامج
  - تنقيح البرنامج
  - اختبار البرنامج

المراجع

- *F. J. Hill & G. R. Peterson*, "Introduction to Switching Theory - Logical Design", John Wiley & Sons, 1980.
- *J.-P. Crestin & D. Jouan*, "Introduction à l'Etude des Systèmes Logiques". ENSTA. 1980.
- *R. E. A. Almain*, "Electronic Logic Systems". Prentice-Hall, 1989.
- *J. M. Trio*, "8086-8088 Architecture and Programming", Macmillan, 1985.
- *R. Dubois*, "Familles 8086-8088 et Z8000 et leurs Coupleurs", Eyrolles, 1985.
- *A. Osborne & G. Kane*, "Osborne 16 bit Microprocessor Handbook", McGraw-Hill, 1981.
- *H. P. Messmer*, "The Indispensable PC Hardware Book", Addison-Wesley, 1995.
- *D. V. Hall*, "Microprocessors and Interfacing - Programming and Hardware", McGraw-Hill, 1986.
- *J. P. Hayes*, "Computter Architecture and Organization", McGraw-Hill, 1986.
- *M. Tischer*, "PC Intern", Abacus, 1992.
- *M. Tischer*, "The PC Bible", MicroApplication, 1996.
- *A. Tanenbaum*, "Structured Computer Organization", Prentice-Hall, 1990.
- *J.-C. Heudin & C. Panetto*, "Les Architectures RISC", Dunod, 1990.
- Intel Databook, "Microprocessors", Vol. 1 & 2, 1991.
- Intel Databook, "Peripheral Components", 1991.
- عوض منصور، *وآخرون*، "البرمجة بلغة التجميع"، شبكة الكمبيوتر الشخصي – الأردن، ١٩٩١.
- محمد بشير *المنجد، وآخرون*، "المدخل إلى المعلوماتية"، جامعة دمشق، ١٩٩٨.
- مروان زبيبي، *وآخرون*، "مبادئ عمل الحواسيب"، جامعة دمشق، ١٩٩٨.
- "كتاب مايكروسوفت لبرمجة المعالجين 80386 و 80486"، الدار العربية للعلوم، ١٩٩١.





مطبعة دار البعث ـ دمشق

سعر المبيع للطالب (١٧٥) ل.س